Discrete Differential Geometry of Thin Materials for Computational Mechanics

Etienne Vouga

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2013

ABSTRACT

Discrete Differential Geometry of Thin Materials for Computational Mechanics

Etienne Vouga

Instead of applying numerical methods directly to governing equations, another approach to computation is to discretize the geometric structure specific to the problem first, and then compute with the discrete geometry. This structure-respecting discrete-differential-geometric (DDG) approach often leads to new algorithms that more accurately track the physically behavior of the system with less computational effort. Thin objects, such as pieces of cloth, paper, sheet metal, freeform masonry, and steel-glass structures are particularly rich in geometric structure and so are well-suited for DDG. I show how understanding the geometry of time integration and contact leads to new algorithms, with strong correctness guarantees, for simulating thin elastic objects in contact; how the performance of these algorithms can be dramatically improved without harming the geometric structure, and thus the guarantees, of the original formulation; how the geometry of static equilibrium can be used to efficiently solve design problems related to masonry or glass buildings; and how discrete developable surfaces can be used to model thin sheets undergoing isometric deformation.

Table of Contents

List of Figures

List of Tables

Acknowledgments

None of this work would have been possible without the mentorship and guidance of my advisor, Eitan Grinspun.

I would also like to thank Glenn Stevens, Ron Goldman, Joe Warren, Max Wardetzky, and Helmut Pottmann, for kindling my interest in mathematics and shaping my career as a scientist; my other coauthors Samantha Ainsley, Basile Audoly, Miklós Bergou, Yotam Gingold, David Harmon, Mathias Höbinger, Danny Kaufman, Scott Schaefer, Breannan Smith, Justin Solomon, Rasmus Tamstorf, and Johannes Wallner; Christopher Batty, Kevin Egan, Akash Garg, Rony Goldenthal, Hao Li, and the other members of the Columbia Computer Graphics Group; those whose funding and software have made my research possible, including the Columbia University School of Engineering, Autodesk, Intel, Google, Nvidia, SideFX, the Walt Disney Company, and the NSF; my thesis proposal and defense committees; and Mom, Maren, Alex, and all other friends and family who have supported me throughout the PhD.

Dedicated to the memory of my grandfathers,
Richard Downs and Paul Vouga,
whose curiosity about the world inspired my love of science.

Chapter 1

Introduction

The influential American modernist architect Louis Sullivan pronounced, at the turn of the 20th century, his now well-known maxim Sullivan [1896]: "It is the pervading law of all things organic and inorganic, ... that form ever follows function. That is the law." While the direction of this causality is today the subject of some debate, there is little dispute that across a wide range of scientific disciplines, from architecture to biology to engineering, form and function are deeply intertwined. Whether a building will stand or fall, why organs and plant structures assume their shape, how skin wrinkles and paper crumples, how well a car's chassis absorbs the impact during a collision – the key to answering all of these questions is the realization that they are, fundamentally, questions about *geometry*.

One of our most powerful tools for understanding physical and biological systems is numerical computation. Gathering meaningful results about a problem using computation is a four-step process:

observed behavior $\Rightarrow$ mathematical model $\Leftrightarrow$ *discretization* of model $\Rightarrow$ computational algorithm

First, the observed phenomenon must be encoded in the language of mathematics, by extracting those physical laws and parameters that are most relevant to the problem. These smooth models must then be discretized: reformulated as a finite-dimensional problem whose solution approximates that of the smooth problem. Numerical algorithms are then applied to the discretization to generate quantitative results.

The standard approach to discretization is to distill the smooth model to a set of PDEs,

which are then approximated using standard numerical techniques such as finite difference, volume, or Galerkin methods. Unfortunately, these PDEs tend to be blind to any geometric structure present in the original problem, and this structure gets lost in translation during discretization. An alternative approach is to *start* by identifying the geometry underlying the smooth model, and discretize the model by discretizing this geometry directly. Respecting the geometry often leads not only to superior, more efficient and more accurate computational methods, but *discrete differential geometry* Bobenko *et al.* [2008] is also a fascinating area of mathematical research on its own, with theorems about discrete geometry often generating new insights about the original smooth geometry.

1.1 Overview and Contributions

Techniques from discrete differential geometry are particularly promising for studying the physics of *thin objects* such as cloth, paper, or freeform masonry or glass surfaces. Even the thinnest of everyday objects, such as a piece of paper, is more than a million atoms thick and could properly be modeled as a volume. However, we usually think of paper as an infinitesimal surface; it is therefore natural to represent such thin objects using simplified geometry. Whereas the differential geometry of smooth surfaces has been thoroughly studied, how to best transfer that understanding to the discrete realm remains an area of active research. This work focuses on problems along three axes:

Structure-preserving time integration with contact Recent work Lew *et al.* [2003] in the field of *geometric time integration* has described how to construct algorithms for simulating deformable bodies like cloth with two characteristic properties: 1) they are *asynchronous*, allowing different forces in the system to act at different, independent rates rather than in lockstep, and 2) ensure *physical correctness*, in terms of provably preserving the conservation laws present in the smooth physics, such as conservation of energy, momentum, and the sympletic form. However, these algorithms are not designed to handle collisions between bodies. Handling collisions is the most difficult part of simulating deformable bodies like pieces of cloth: forces arising from collisions are *non-local* and *stiff*: the forces must be strong enough to keep the cloth from passing through itself no matter

how strongly it is being pulled or squeezed.

Chapter 2 discusses how the ideas of asynchronous geometric time integration can be modified to handle the contact forces that occur during cloth simulation. A proof is presented that these modifications do not compromise the guaranteed preservation of physical invariants.

Although the method described in Chapter 2 comes with unparalleled robustness and correctness guarantees, the original, straightforward implementation was not computationally efficient. Chapter 3 combines asynchronous integration of contact with *speculative simulation and rollback* to increase the efficiency of the method by over two orders of magnitude, without affecting the accuracy or any of the theoretical guarantees.

Stability of self-supporting masonry structures Masonry structures built out of materials like stone, brick, or unreinforced concrete are extremely strong under compression, but are held together by friction or mortar with low tensile strength. These extreme material properties imply that the stability of masonry structures can be analyzed from their *shape* alone, without worrying about material failure. Moreover, while we usually think of buildings as volumetric structures, the problem of stability of such structures can be reduced to that of surfaces. The physics and engineering problem of analyzing and desiging stable masonry structure is therefore at heart a surface geometry problem.

Chapter 4 discusses the rich connections between the static equilibrium of surfaces and diverse topics in differential geometry, discrete geometry, and mechanics. These insights lead to an interactive tool for designing self-supporting surfaces that is an order of magnitude faster than prior work; an algorithm for finding planar quadrilateral remeshings of self-supporting surfaces that are still stable; and the characterization of special one-parameter families of surfaces all of whose members are guaranteed to be stable.

Dynamics of developable surfaces The deformation of a thin, flat piece of material, such as a sheet of paper or a soup can, as it is being crumpled is governed by competition between the material's bending and stretching energies. A thin surface bends much more readily than it stretches: for instance crumpling a sheet of paper adds wrinkles and folds but does not noticeably affect its rectangular intrinsic shape. This observation, supported

by results from mechanics that stress concentrates along ridges, suggests that the crumpling and buckling of an initially flat thin shell could be modeled as an isometric deformation of a *discrete developable surface*. Simulating crushing and crumpling using such a reduced representation has several advantages over straightforward finite element simulations of the surface: i) a discrete developable surface is designed to allow the sharp creases that arise during crumpling; ii) a surface that bends and creases but does not stretch can be accurately simulated using much larger, computationally-efficient elements; iii) *membrane locking*, where the stretching forces artificially interfere with the surface's ability to bend, is sidestepped. Chapter 5 describes preliminary investigation of this idea.

1.2 Background and Related work

1.2.1 Structured Integration

Variational integrators (VIs) Suris [1990]; MacKay [1992]; Marsden and West [2001] are a general class of time integration methods for Hamiltonian systems whose construction guarantees certain properties highly desirable of numerical simulations. Instead of directly discretizing the smooth equations of motion of a system, the variational approach instead discretizes the system's action integral. By analogy to Hamilton's least action principle, a discrete action can be formed, and *discrete* Euler-Lagrange equations derived by examining paths which extremize it. From the Euler-Lagrange equations, discrete equations of motion are readily recovered. As a consequence of this special construction, VIs are guaranteed to satisfy a discrete formulation of Noether's Theorem West [2004], and as a special case conserve linear and angular momentum. VIs are automatically *symplectic* Hairer *et al.* [2006]; while they do not necessarily conserve energy, conservation of the symplectic form assures no-drift conservation of energy over exponentially many time steps Hairer *et al.* [2006].

1.2.2 Contact

The simplest contact models for finite element simulation follow the early analytical work of Hertz 1882 in assuming frictionless contact of planar (or nearly planar) surfaces with small

strain. In this regime, several approaches have been explored to arrive at a weak formulation of contact; for a high-level survey of these approaches, see for example the overview by Belytschko et al 2006 or Wriggers 1995. The first of these are the use of penalty forces, described for instance by Oden 1980 and Kikuchi and Oden 1988. The penalty approach results in a contact force proportional to an arbitrary *penalty stiffness* parameter and to the rate of interpenetration, or in more general formulations to an arbitrary function of rate of interpenetration and interpenetration depth; Belytschko and Neal 1991 discuss the choosing of this parameter in Section 8. Recent work by Belytschko et al 2002 uses moving least squares to construct an implicit smooth contact surface, from which the interpenetration distance is evaluated. Peric and Owen 1992 describe how to equip penalty forces with a Coulomb friction model.

Seeking to exactly enforce non-penetration along the contact surface leads to generalizations of the method of Lagrange multipliers. Hughes et al 1976 and Nour-Omid and Wriggers 1986 provide an overview of this approach in the context of contact response. Such contraint enforcement can be viewed as a penalty force in the limit of infinite stiffness, impossible to attain in practice since the system becomes ill-conditioned. Taylor and Papadopoulos 1993 considers persistent contact by extending Newmark to treat jump conditions in kinematic fields, thus reducing undesirable oscillatory modes. However, the effects of these modifications on numerical dissipation and long-time energy behavior is not considered.

The Augmented Lagragian method blends the penalty and Lagrange multiplier approaches, and combines the advantages of both: unlike for pure penalty forces, convergence to the exact interpenetration constraint does not require taking the penalty stiffness to infinity, and the Lagrange multiplier solve tends to be well-conditioned. Bertsekas 1984 gives a mathematical overview of the augmented Lagrangian method, and Wriggers et al 1985 and Simo and Laursen 1992 expand on its application to contact problems in finite elements.

Non-smooth contact requires special consideration, since in the non-smooth regime there is no straightforward way of defining a contact normal or penetration distance. Simo et al 1985 discretize the contact surface into segments over which they assume constant contact pressure; this formulation allows them to handle non-node-to-node contact using

a perturbed Lagrangian. Kane et al 1999 apply non-smooth analysis to resolve contact constraints between sharp objects. Pandolfi et al 2002 extend the work of Kane et al by describing a variational model for non-smooth contact with friction. Cirak and West 2005 decompose contact resolution into an impenetrability-enforcement and momentum-transfer step, thereby exactly enforcing non-interpenetration while nearly conserving momentum and energy.

1.2.2.1 Structured Integration of Contact

Given the many advantages of VIs, it is natural to apply them to the handling of contact and impact, a long-studied and challenging problem in physical simulation. Unfortunately, a naïve application of a contact algorithm to a variational integrator is not guaranteed to preserve the variational structure of the time integration method, and in practice one observes that the good energy behavior is lost. For this reason, several authors have explored a structure-preserving approach to solving the contact problem. Barth et al 1999 consider an adaptive-step-size algorithm that preserves the time-reversible symmetry of the RATTLE algorithm, and demonstrate an application to an elastic rod interacting with a Lennard-Jones potential. Kane et al 2000 show that the Newmark method, for all parameters, is variational, and construct two two-step dissipative integrators that yield good energy decay. Laursen and Love 2002, by taking into account velocity discontinuities that occur at contact interfaces, develop a momentum- and energy-preserving method for simulating frictionless contact.

Common to all these approaches is a *synchronous* treatment of global time, in which the entire configuration is advanced from one intant in time to the next. While synchronous integration is attractive for its simplicity, it has the drawback that a spatially-localized stiff mode—such as that associated to a localized contact—can force the global configuration to advance at fine time steps.

Indeed, mechanical systems are almost never uniformly stiff. Different potentials have different stable time step requirements, and even for identical potentials this requirement depends on element size, since finer elements can support higher-energy modes than coarser elements. Any global time integration scheme cannot take advantage of this variability, and

instead must integrate the entire system at the globally stiffest time step.

1.2.3 Asynchronous Variational Integrators

Suppose the system can be partitioned into elements such that each force acts entirely within one element. Then *asynchronous variational integrators* (AVIs) Lew *et al.* [2003] generalize VIs by allowing each element to have its own, independent time step. Coarser elements can then be assigned a slower "clock," and finer elements a faster one. Asynchrony avoids the undesirable situation in which a small number of very fine elements degrade overall performance. AVIs retain all of the properties of variational integrators mentioned above, except for discrete symplecticity. However, AVIs instead preserve an analogous *discrete multisymplectic* form, and it has been shown experimentally that preservation of this form likely induces the same long-time good energy behavior that characterize symplectic integrators Lew *et al.* [2003]. Parallel extensions of AVIs have been studied for use in finite element simulations of elastica without contact, using domain decomposition and message passing Kale and Lew [2007] as well as dependency graphs Huang *et al.* [2007]. An implementation of the latter has been incorporated into the Galois framework for running parallel algorithms on multiprocessors Pingali *et al.* [2011].

In graphics, AVIs have been applied to asynchronous integration of cloth bending and stretching forces Thomaszewski *et al.* [2008b]. Debunne et al 2001 proposed the similar idea of adaptively switching between different levels of resolution and time step when simulating visco-elastic bodies. My collaborations on asynchronous contact mechanics for cloth simulation Harmon *et al.* [2009, 2012]; Vouga *et al.* [2011]; Ainsley *et al.* [2012] were the first to consider an asynchronous, variational treatment of contact, and it will be shown in Chapter 2 that it retains multisymplecticity. These ideas were recently extended to handle implicit forces using a ghost mesh Harmon *et al.* [2011]. Rangarajan et al. 2008 suggest AVIs for simulating penetration of a soft hyperelastic material by rigid bodies, and propose handling contact by reflecting momentum at the end of any elemental time step during which contact occurred. This method was observed to dissipate energy during contact events; the amount of drift can be controlled by appropriately decreasing the time steps of elements involved in contact. Ryckman and Lew 2011 concurrently investigated extending the AVI

framework to incorporate contact response.

1.2.4 Time Warp

Instead of predicting and preventing collisions by looking ahead at the future of a simulation, the alternate *time warp* paradigm Jefferson [1985] makes no attempt to predict collisions in advance; instead, it advances time and then checks for collisions *in hindsight*, rewinding time and correcting collisions if any are detected. Mirtich 2000 applied this Time Warp idea to simulations of large numbers of interacting rigid bodies, and our method is largely inspired by this work. Zheng and James 2011 recently used Time Warp at the body level to asynchronously simulate vibrating objects for sound simulation. These methods leverage Time Warp to roll back only those contact groups involved in a detected collision; for simulations with few, large, deformable bodies this approach is less profitable since stiff elastic forces rapidly propagate information away from points of contact. Rather, in Chapter 3 we use Time Warp primarily because of the substantial savings Retroactive Detection offers asynchronous integration.

1.2.5 Parallel Cloth Simulation

Parallel simulation of cloth and thin shells with collisions is a well-studied problem. For instance, Thomaszewski 2006; 2008a solves for implicit material forces using a data-parallel conjugate gradient algorithm, and uses data from past frames to estimate a good splitting of the collision detection task. Bender and Bayer 2008 simulate inextensible cloth by decomposing it into strips of constraints that can be processed in parallel. Selle et al 2009 efficiently handle very high resolution cloth by parallelizing and extending Bridson's method 2002 to reduce the number of geometric tests needed during collision detection.

1.2.6 Equilibrium of Masonry Structures

Unsupported masonry has been an active topic of research in the engineering community. The foundations for the modern approach were laid by Jacques Heyman 1966 and are available as a textbook Heyman [1995]. The theory of reciprocal force diagrams in the planar case was studied by J. Maxwell; a unifying view on polyhedral surfaces, compressive forces

and corresponding "convex" force diagrams is presented by Ash *et al.* [1988]. F. Fraternali 2002; 2010 established a connection between the continuous theory of stresses in membranes and the discrete theory of forces in thrust networks, by interpreting the latter as a non-conforming finite element discretization of the former.

Several authors have studied the problem of finding discrete compressive force networks contained within the boundary of masonry structures; previous work in this area includes O'Dwyer [1998] and Andreu *et al.* [2007]. Fraternali 2010 proposed solving for the structure's discrete stress surface, and examining its convex hull to study the structure's stability and susceptibility to cracking. This approach works well for analyzing existing structures, where the boundary tractions can be measured and the stress surface is known to be close to convex, but is not an ideal design tool since in such settings the boundary tractions are unknown, and where replacing a non-convex initial stress surface by its convex hull can cause large, uncontrolled global changes to the surface being designed.

1.2.6.1 Thrust Network Analysis

Philippe Block's seminal thesis introduced *Thrust Network Analysis*, which pioneered the use of thrust networks and their reciprocal diagrams for efficient and practical design of self-supporting masonry structures. By first seeking a reciprocal diagram of the top view, guaranteeing equilibrium of horizontal forces, then solving for the heights that balance the vertical loads, Thrust Network Analysis linearizes the form-finding problem. For a thorough overview of this methodology, see e.g. Block and Ochsendorf [2007]; Block [2009]. Recent work by Block and coauthors extends this method in the case where the reciprocal diagram is not unique; for different choices of reciprocal diagram, the optimal heights can be found using the method of least squares Van Mele and Block [2011], and the search for the best such reciprocal diagram can be automated using a genetic algorithm Block and Lachauer [2011].

1.2.6.2 Other Form-Finding Paradigms

Other approaches to the design of self-supporting structures include modeling these structures as damped particle-spring systems, with loads applied to the particles ("dynamic

relaxation" methods) Kilian and Ochsendorf [2005]; Barnes [2009], and mirroring the rich tradition in architecture of designing self-supporting surfaces using hanging chain or membrane models (for instance by Frei Otto, Antoni Gaudi, and Heinz Isler) Heyman [1998]; Kotnik and Weinstock [2012]. Force density methods Linkwitz and Schek [1971] linearize the form-finding problem by solving for static equilibrium with respect to position variables, given prescribed prestresses in the form of axial force densities Gründig *et al.* [2000]. Alternatively, masonry structures can be represented by networks of rigid blocks Livesley [1992], whose conditions on the structural feasibility were incorporated into procedural modeling of buildings Whiting *et al.* [2009, 2012].

1.2.7 PQ Meshes and Conjugate Curve Networks

Algorithmic and mathematical methods relevant to this proposal are work on the geometry of PQ meshes Liu *et al.* [2006a], discrete curvatures for such meshes Pottmann *et al.* [2007]; Bobenko *et al.* [2010], in particular curvatures in isotropic geometry Pottmann and Liu [2007]. Schiftner and Balzer 2010 discuss approximating a reference surface by a quad mesh with planar faces, whose layout is guided by statics properties of that surface.

1.2.8 Physics of Elastic Surfaces

That stretching energy of a developable surface concentrates at singular ridges and cone points (d-cones) has been observed in physical experiments and simulations Boudaoud *et al.* [2000]; DiDonna [2002] and analyzed using scans of crumpled paper Blair and Kudrolli [2005]. Scaling laws have been derived Lobkovsky and Witten [1996]; Cerda and Mahadevan [1998]; Venkataramani [2004]; Witten [2006] for this localized energy, as will be revisited in Chapter 5. Along these lines, Cerda and Mahadevan 2003 developed scaling laws for the wavelengths of wrinkles in elastic sheets.

It has been shown that in the limit of infinite thinness, the surface is perfectly developable, and assumes a Yoshumura "diamond" pattern Yoshimura [1955]; Hoff *et al.* [1966]. Ben Amar and Pomeau 1997 consider developable surfaces bounded by given curves, and show that the resulting surface is not generally smooth. The statistics of the lengths of creases Sultan and Boudaoud [2006] have also been studied. Unfortunately, it is not gen-

erally true that a buckled elastic sheet (away from the limit of infinitesimal thickness) is developable almost everywhere – a small, diffuse amount of stretching (measurable as non-vanishing Gaussian curvature) is essential to the observed behavior of some buckled materials, such as the pinched pipe Mahadevan *et al.* [2007]. The extent to which a buckled elastic sheet exhibits localized versus diffuse stress is not yet completely understood, although experiments on confined sheets suggest Schroll *et al.* [2011] that the sheet can be partitioned into regions governed by one behavior or the other.

1.2.9 Discrete Developable Surfaces

Early work Kergosien *et al.* [1994] modeled developable patches as a pair of parameterized boundary curves, and presented an algorithm for permitting creasing by adding an additional interior boundary curve. Other non-mesh representations of discrete developable surfaces include piecewise generalized cones Redont [1989]; Sun and Fiume [1996] and cone splines Leopoldseder and Pottmann [1998]; Chen *et al.* [1999], a geodesic curve or "spine" through the surface Bo and Wang [2007], a Bezier patch passing through a given curve, networks of curved triangular patches Chen and Tang [2013], or constructed using a deCasteljau-style algorithm Chu and Squin [2002]; Aumann [2003, 2004].

A common mesh representation of a developable surface is as a triangle mesh with edge-length constraints enforced either by Lagrange multipliers, or penalty forces Chen and Tang [2010]; for a prescribed set of fixed vertices it is possible to also represent the constrained mesh in reduced coordinates Liu *et al.* [2007a]. The dynamics of these inextensible triangle meshes have been simulated Liu *et al.* [2007a]; Chen and Tang [2010], but avoiding membrane locking requires higher-order elements, non-conforming elements English and Bridson [2008], or adaptivity. Arbitrary Lagrangian-Eulerian (ALE) methods are commonly used for adaptive finite element simulations; see for instance Donea *et al.* [2004]. Such remeshing has also been studied in the context of structured integration Mosler and Ortiz [2006]; Lahiri *et al.* [2008]. An alternative to adding material degrees of freedom is frequent remeshing of the material domain, which has been exploited recently in graphics for simulating cloth and paper Narain *et al.* [2012, 2013]; problems inherent to remeshing remain, such as the introduction of popping artifacts and drift of physical invariants.

For simulating thin sheets, *nearly*-developable surfaces are perhaps a better representation than perfectly developable ones. Nearly-developability is also of interest in the context of mesh parameterization, which seeks to cut surfaces into patches which can be developed with minimal area distortion Julius *et al.* [2005]; Wang [2008]. Nearly-developable surfaces also arise from some algorithms designed to find developable surfaces, when the boundary conditions do not admit perfectly developable solutions Pérez and Suárez [2007]; Tang and Chen [2009]. Formal study of quasi-developable surfaces, and of kinematic discretizations of such surfaces, remains largely open.

The discrete representation presented in Chapter 5 builds on earlier work parameterizing such surfaces with discrete rules Perriollat [2007] and a previous collaboration on interactive modeling of such surfaces Solomon *et al.* [2012].

1.2.9.1 From Boundary Curves

There are many techniques for generating developable surfaces given presribed boundary curve or curves, both of which are generally underconstrained problems, especially if the surface is permitted to have interior d-cones Frey [2004]. For a single curve, the method of Frey 2002 samples the curve and looks at all possible triangulations of those samples, choosing the one that would result from a piece of paper being crushed between two approaching parallel copies of the boundary polygon. Rohmer et al 2011 use a divide-and-conquer approach based on finding fold lines that split the bounding curve into two.

For a pair of boundary curves, early work by Weiss and Furtner 1988 proposed marching around the boundaries to find planar strips connecting them, thus implicitly finding the rulings connecting one boundary to the other. This idea has been extended Pérez and Suárez [2007] to algorithms that first find rulings spanning the curves and then lofted to B-spline surfaces. The boundary of the convex hull of any number of curves is always a piecewise developable surface, and algorithms have been developed Rose *et al.* [2007] for generating developable surfaces by choosing from these pieces. Recent work by Chen and Tang 2013 interpolates given curves by G^1 networks of non-planar triangular patches.

1.2.9.2 Design

New discrete approximations of developable surfaces can be generated through subdivision of planar quadrilateral control strips Liu *et al.* [2006a], manipulating control curves that become geodesic curves of the surface Bo and Wang [2007], etc.

Developable surfaces can also be designed by starting from a nondevelopable surface, and projecting it, in some sense, onto the space of developable ones. The most straightforward approach to doing so is to enforce the constraint of zero Gaussian curvature, which can be discretized as zero angle deficit around each interior vertex; unfortunately, doing so requires solving a global, non-convex optimization problem and is computationally expensive. Tang and Chen 2009 describe a framework for performing this optimization subject to additional terms that enforce fitting of the surface to anchor positions and minimize distortion of the original surface. Wang 2004 proposes instead a Gauss-Seidel-style iterative method where each vertex in turn is allowed to move in its normal direction to minimize the angle deficit of the vertex and its neighbors. Liu et al 2007b approximates a rectangular surface patch by developable strips, with the strips arranged to minimize approximation error, reminiscent of Weiss and Furtner's general approach. Chen et al 1999 fits a developable surface represented as a cone spline to input data by fitting cones of revolution to the data.

Starting with a mesh in the plane, and an initial noncorming, isometric embedding of its faces into $\mathbb{R}^3$, a developable surface can be found approximating the initial embedding by simulatenously optimizing for planar and spatial positions that minimizes vertex disagreement in space, and some number of additional fitting or fairness energy terms. Kilian, Mitra, and coauthors Kilian *et al.* [2008]; Mitra *et al.* [2008] apply this approach to fit developable quad surfaces to arbitrary input surfaces.

1.2.9.3 As Origami Models

Discrete developable surfaces are also intimately related to the study and design of origami patterns. Burgoon et al 2006 propose a framework for interactively folding origami using a stiff thin shell simulation, along with extra tools for adding fold edges and pinning parts of the paper; exact developability is not enforced. The rigidity of valence-four origami has been extensively analyzed Tachi [2009a] and bears on the question of membrane locking

of discrete developable surfaces. Tachi 2009b; 2010a described a system for interactively editing origami by drawing crease line and manipulating angles, with integrability constraints around vertices as described in Chapter 5, and discusses how to adjust crease lines to prevent locked interior vertices that would otherwise appear where folds intersect.

Work in this direction that does not directly relate to the representation of discrete developable surfaces or the simulation of thin sheets includes computing developments of arbitrary polyhedral surfaces Shinagawa *et al.* [2002], mesh simplification to ease this cutting process Mitani and Suzuki [2004], formalizing origami fold operations as transformations of abstract graphs Ida [2008]; Ida and Takahashi [2010], constructing origami approximations of any arbitrary (curved) surface using a system of tucks Tachi [2010b], designing networks of thick panels with the same flexibility as infinitesimal origami Tachi [2011], and study of the mechanical properties of thin shells folded into origami patterns Schenk and Guest [2011].

Part I

Geometric Integration of Deformable Bodies in Contact

Chapter 2

Asynchronous Contact Mechanics

Handling collisions is one of the hardest parts of simulating a physical system, for several reasons: firstly, it is nearly impossible to tell from initial conditions which objects will end up colliding and which ones will not. All eventualities must therefore be taken into account, at great expense. Secondly, the forces needed to resolve collisions are very stiff : to stop two fast-moving objects from interpenetrating, a large force needs to be applied to the two objects over a short amount of time. Applying such extreme forces without destabilizing the system and injecting into it large error requires care. Lastly, should any error occur during collision response, the consequences are often dramatic and catastrophic: for example, a ball that is completely enclosed inside a box might tunnel through one of the walls.

We can identify those invariants present in a physical simulation with collisions, each of which will become a key property a satisfactory simulation algorithm for contact must possess:

1. *Safety*: Objects never interpenetrate or tunnel through each other. That is, all collisions are correctly resolved.

2. *Correctness*: Momentum, angular momentum, and energy are conserved.

Ensuring both of these properties is not straightforward: the amount of force that must be applied to a pair of objects to stop them from interpenetrating increases the greater the velocity at which they are approaching: faster-moving objects are harder to divert.

Finitely-stiff forces thus are not guaranteed to stop all collisions, precluding the use of standard structure-preserving algorithms. A more sophisticated approach is needed.

In addition to safety and correctness, several other properties and features are required of any practical collision simulation. These include:

3. *Progress*: A well-posed simulation should finish in a finite amount time, i.e., should require only a finite amount of computation to simulate. A simulation that does not finish is simply impractical.

4. *Dissipation and Friction*: A ball dropped onto a floor does not bounce back quite to its original height. This phenomenon is one example of energy dissipation that often occurs when real-world objects collide. Another is friction. Although a simulation algorithm should conserve energy in the absence of these phenomena, it should be possible to add them to the system, in which case the energy should decay in a graceful, controlled manner over time.

5. *Efficiency*: The progress property is a bare minimum requirement for an algorithm to be practical; ideally, the algorithm should possess all of the above-listed properties, and perform with efficiency comparable to current state-of-the-art methods.

In my work Vouga *et al.* [2011] and that of my collaborators Harmon *et al.* [2009]; Harmon [2010], we describe the first algorithm for simulating thin, deformable objects like cloth undergoing contact that is guaranteed to satisfy the first three properties: i) all collisions are prevented; ii) physical invariants like momentum and energy are conserved; and iii) the simulation finishes in finite time. These guarantees hinge on the fact that the algorithm is built on an *asynchronous, multisymplectic integrator*, extending Lew et al's investigation of asynchronous variational integrators 2003, that can handle arbitrary, spatially non-local potentials each advancing at their own independent time step. In this chapter, I will describe this integrator and prove that it possesses these properties; I will also summarize how to build an algorithm for asynchronous contact mechanics (ACM) with the three guarantees on top of this integrator. More details on the latter can be found in Harmon's thesis 2010 and my collaboration with Harmon 2009. Some preliminary work on

incorporating dissipation and friction into ACM will also be described. Chapter 3 will focus on the last property, efficiency.

2.1 Background

The starting point for building the ACM algorithm is the selection of the penalty method as a model for contact Wriggers [2002]; Laursen [2002]. For each pair of elements in the system, a potential is added that is (piecewise) quadratic in the *gap function* measuring the separation distance between the two elements. This potential vanishes when elements are sufficiently far apart, and increases with increasing interpenetration, so that approaching elements feel a force that resists impact. This approach suffers two limitations, however. Firstly, these contact potentials are fundamentally nonlocal phenomena: for every pair of elements that might come into contact during the course of the simulation, a potential coupling the two must be added. As will be shown, the fact that contact potentials cannot be expressed as the integration over the material domain of an energy density depending only on a neighborhood of the domain will present a technical obstruction to the original formulation of AVIs, but fortunately one that can be overcome by a natural generalization.

Secondly, penalty forces have a well-studied performance-robustness tradeoff Belytschko and Neal [1991]: adding a half-quadratic potential requires choosing an arbitrary stiffness parameter, and for any stiffness chosen for the penalty potential, two approaching elements will interpenetrate some distance, and in the worst case *tunnel* completely through each other. Moreover, the stable time step of the penalty force decreases as stiffness increases, so choosing a very stiff penalty potential is untenable as a solution to excessive penetration or tunneling. In practice, users of the method must determine an adequate penalty stiffness by iterated tweaking of parameters, until the simulation completes without collision artifacts. An appealing modification of the penalty approach replaces the quadratic potential with a nonlinear *barrier potential* Nocedal and Wright [2000] that diverges as the configuration approaches contact. Because the barrier diverges, its stiffness is unbounded, necessitating a time-adaptive time stepping method. Our method uses a discrete analogue of the barrier potential—an infinite sequence of *discrete penalty layers*—that in effect enables AVIs to

serve as adaptive integrators.

The remainder of this chapter

- reviews the concepts underlying variational integrators (section 2.3);
- extends the construction of AVIs so that, unlike in previous work, a discretization into disjoint elements is not necessary, by associating a clock to each force instead of to each element (section 2.4);
- demonstrates that this generalization does not destroy the desirable integration properties guaranteed by the variational paradigm, most importantly the conservation of the discrete multisymplectic form (section 2.4.1);
- briefly summarizes how to leverage this extension to equip the AVI framework with a contact model. The proposed barrier method uses a divergent sequence of quadratic potentials that guarantees non-penetration and retains the asynchrony or conservation properties of AVIs (section 2.5);
- provides some implementation details for efficiently simulating this contact model (section 2.6). Moving planar slabs, an example of a *kinetic data structure*, are used to conservatively estimate when to begin integrating each quadratic potential;
- describes some mechanisms for introducing controlled dissipation into the contact model (section 2.7);
- presents representative examples and discusses future work (section 2.8).

2.2 Contributions

The novel time integrator presented in section 2.4 extends the theory of asynchronous variational integration Lew *et al.* [2003] to handle the inclusion of arbitrary, non-local forces, and the first part of the chapter describes this extension and proves its correctness. Experimental validation, in chapter 2.8 and in my accompanying publication Vouga *et al.* [2011], demonstrates the expected good, long-term energy behavior of the method. One notable application of this integrator was to the construction Harmon *et al.* [2009]; Harmon [2010]

of a method for stably and adaptively simulating thin objects subject to self-contact, using the aforementioned integrator and a novel decomposition of the contact potentials into *spatially adaptive* piecewise quadratic pieces (see section 2.5).

2.3 Variational Integrators

This section presents background on variational integration and symplectic structure Hairer *et al.* [2006]; Marsden and West [2001]; West [2004].

Let $\gamma(t)$ be a piecewise-regular trajectory through configuration space $\mathbf{Q}$, and $\dot{\gamma}(t) = \frac{d}{dt}\gamma(t)$ be the configurational velocity at time t. For simplicity, assume that the kinetic energy of the system T depends only on configurational velocity, and that the potential energy V depends only on configurational position, so that the Lagrangian L at time t may be written as

$$L(q, \dot{q}) = T(\dot{q}) - V(q). \tag{2.1}$$

Then given the configuration of the system q_0 at time t_0 and q_f at t_f, Hamilton's principle Lanczos [1986] states that the trajectory of the system $\gamma(t)$ joining $\gamma(t_0) = q_0$ and $\gamma(t_f) = q_f$ is a stationary point of the action functional

$$S(\gamma) = \int_{t_0}^{t_f} L\left[\gamma(t), \dot{\gamma}(t)\right] dt$$

with respect to taking variations $\delta\gamma$ of γ which leave γ fixed at the endpoints t_0, t_f. In other words, γ satisfies

$$dS(\gamma) \cdot \delta\gamma = 0. \tag{2.2}$$

Integrating by parts, and using that $\delta\gamma$ vanishes at t_0 and t_1,

$$\begin{aligned} dS(\gamma) \cdot \delta\gamma &= \int_{t_0}^{t_f} \left(\frac{\partial L}{\partial q}(\gamma, \dot{\gamma}) \cdot \delta\gamma + \frac{\partial L}{\partial \dot{q}}(\gamma, \dot{\gamma}) \cdot \delta\dot{\gamma} \right) dt \\ &= \int_{t_0}^{t_f} \left(-\frac{\partial V}{\partial q}(\gamma) - \frac{\partial^2 T}{\partial \dot{q}^2}(\dot{\gamma})\ddot{\gamma} \right) \cdot \delta\gamma \, dt = 0. \end{aligned}$$

Since this equality must hold for all variations $\delta\gamma$ that fix γ's endpoints,

$$\frac{\partial V}{\partial q}(\gamma) + \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}(\dot{\gamma}) \right) = 0, \tag{2.3}$$

the *Euler-Lagrange equation* of the system. This equation is a second-order ordinary differential equation, and so has a unique solution γ given two initial values $\gamma(t_0)$ and $\dot{\gamma}(t_0)$.

2.3.1 Symplecticity

The flow $\Theta_s : [\gamma(t), \dot{\gamma}(t)] \mapsto [\gamma(t+s), \dot{\gamma}(t+s)]$ induced by (2.3) has many structure-preserving properties; in particular it is momentum-preserving, energy-preserving, and symplectic Lew [2003]. To derive this last property, for the remainder of this section the space of trajectories is restricted to those that satisfy the Euler-Lagrange equations. For such trajectories, if the requirement that $\delta\gamma$ fix the endpoints of γ is relaxed, then the boundary terms of the integration by parts are no longer 0 and

$$dS(\gamma) \cdot \delta\gamma = \frac{\partial T}{\partial \dot{q}} \left[\pi_{\dot{q}}(q, \dot{q})\right] \cdot \delta\gamma \Big|_{t_0}^{t_f}, \tag{2.4}$$

where $\pi_{\dot{q}}$ is projection onto the second factor.

Since initial conditions $(q, \dot{q})$ are in bijection with trajectories satisfying the Euler-Lagrange equation, such trajectories γ can be uniquely parametrized by initial conditions $[\gamma(t_0), \dot{\gamma}(t_0)]$. For the remainder of this section variations $\delta\gamma$ are also restricted to *first variations*: those variations in whose direction γ continues to satisfy the Euler-Lagrange equations. These are also parametrized by variations of the initial conditions, $(\delta q, \delta\dot{q})$. For conciseness of notation, the change of variables $\nu(t) = (\gamma(t), \dot{\gamma}(t))$ and $\delta\nu(t) = [\delta\gamma(t), \delta\dot{\gamma}(t)]$ can be used; using this notation the above two facts can be rewritten as $\nu(t) = \Theta_{t-t_0}\nu(t_0)$ and $\delta\nu(t) = \Theta_{t-t_0 *}\delta\nu(t_0)$. The action (2.1), a functional on trajectories γ, can also be rewritten as a function S_i of the intial conditions,

$$S_i(q, \dot{q}) = \int_0^{t_f - t_0} L\left[\Theta_t(q, \dot{q})\right] dt,$$

so that

$$dS(\gamma) \cdot \delta\gamma = dS_i\left[\nu(t_0)\right] \cdot \delta\nu(t_0).$$

Substituting all of these expressions into (2.4),

$$\begin{aligned} dS_i\left[\nu(t_0)\right]\cdot\delta\nu(t_0) &= \left(\frac{\partial T}{\partial \dot{q}}\circ\pi_{\dot{q}}\right)\left[\Theta_{t-t_0}\nu(t_0)\right]\cdot\delta\gamma(t)\Big|_{t_0}^{t_f} \\ &= \left(\frac{\partial T}{\partial \dot{q}}\circ\pi_{\dot{q}}\right)\left[\Theta_{t-t_0}\nu(t_0)\right]dq\cdot\delta\nu(t)\Big|_{t_0}^{t_f} \\ &= \left(\frac{\partial T}{\partial \dot{q}}\circ\pi_{\dot{q}}\right)\left[\Theta_{t-t_0}\nu(t_0)\right]dq\cdot\Theta_{t-t_0*}\delta\nu(t_0)\Big|_{t_0}^{t_f} \\ &= \left(\Theta_{t_f-t_0}{}^{*}\theta_L-\theta_L\right)_{\nu(t_0)}\cdot\delta\nu(t_0), \end{aligned}$$

where θ_L is the one-form $\left(\frac{\partial T}{\partial \dot{q}}\circ\pi_{\dot{q}}\right)dq$. Since dS_i is exact,

$$d^2S_i = 0 = \Theta_{t_f-t_0}{}^{*}d\theta_L - d\theta_L,$$

so since t_0 and t_f are arbitrary, $\Theta_s^* d\theta_L = d\theta_L$ for arbitrary times s, and Θ preserves the so-called *symplectic form* $d\theta_L$.

2.4 Asynchronous Variational Integrators

Discrete mechanics Veselov [1988]; Suris [1990]; Moser and Veselov [1991]; Wendlandt and Marsden [1997]; Marsden and West [2001]; Hairer *et al.* [2006] describes a discretization of Hamilton's principle, yielding a numerical integrator that shares many of the structure-preserving properties of the continuous flow Θ_s. In particular, asynchronous variational integrators, introduced by Lew et al 2003, are a family of numerical integrators, derived from a discrete Hamilton's principle, that support integrating potentials at different time steps. Their formulation assumes a spatial partition, with each potential depending only on the configuration of a single element; in this exposition, the general arguments set forth by Lew et al are followed, but the notation and derivation departs from their work as necessary to support potentials with arbitrary, possibly non-disjoint spatial stencil.

Let $\{V^i\}$ be potentials with time steps h^i. Each potential V^i is concerned with certain moments in time—namely, integer multiples of h^i—and these moments are inconsistent across different potentials. Time is therefore subdivided in a way compatible with all

potentials: for a τ-length interval of time, the set $\Xi(\tau)$ is defined by

$$\Xi(\tau) = \bigcup_{V^i} \bigcup_{j=0}^{\lfloor \tau/h^i \rfloor} jh^i.$$

That is, $\Xi(\tau)$ is the set of all integer multiples less than τ of all time steps. Ξ can be ordered, and in particular let $\xi(i)$ be the $(i+1)$-st least element of Ξ. For ease of notation, also let $\omega^i(j) = \xi^{-1}(jh^i)$; that is, ω converts the jth timestep of potential i into a global time index.

If n is the cardinality of Ξ, a trajectory of duration τ is then discretized by linearly interpolating intermediate configurations $q_0, q_1, \ldots, q_{n-1}$, where q_i is the configuration of the system at time $\xi(i)$. Velocity is discretized as $\dot{q}_{k+1/2} = \frac{q_{k+1}-q_k}{\xi(k+1)-\xi(k)}$ on the segment of the trajectory between q_k and q_{k+1}. A global action functional of these trajectories is needed, and can be constructed in the natural way:

$$S_g(\mathbf{q}) = \sum_{j=0}^{n-2} \left[\xi(j+1) - \xi(j)\right] T_d \left[q_j, q_{j+1}, \xi(j), \xi(j+1)\right] - \sum_{V^i} \sum_{j=1}^{\lfloor \tau/h^i \rfloor} h^i V^i(q_{\omega^i(j)}),$$

where, for $T(\dot{q})$ the kinetic energy of the entire configuration, $T_d(q_a, q_b, t_a, t_b) = T\left(\frac{q_b-q_a}{t_b-t_a}\right)$. For use in the following, also let $T'_d(q_a, q_b, t_a, t_b) = \frac{\partial T}{\partial \dot{q}}\left(\frac{q_b-q_a}{t_b-t_a}\right)$.

No attempt has been made to define a Lagrangian pairing the kinetic and potential energy terms; it will be seen that an action defined in this way still leads to a multisymplectic numeric integrator. To this end, Hamilton's principle $dS_g(\mathbf{q}) \cdot \delta\mathbf{q} = 0$ is imposed for variations $\delta\mathbf{q} = \{\delta q_0, \ldots, \delta q_{n-1}\}$ with $\delta q_0 = \delta q_{n-1} = 0$. Then S_g can be rewritten as

$$S_g(\mathbf{q}) = \sum_{j=0}^{n-2} \left[\xi(j+1) - \xi(j)\right] T_d \left[q_j, q_{j+1}, \xi(j), \xi(j+1)\right] - \sum_{j=1}^{n-1} \sum_{h^i | \xi(j)} h^i V^i(q_j), \tag{2.5}$$

where the notation $h^i|\xi(j)$ is abused to mean "all indices i for which h^i evenly divides $\xi(j)$,"

so that

$$\begin{aligned} dS_g(\mathbf{q}) \cdot \delta\mathbf{q} &= \sum_{j=0}^{n-2} T_d'\left[q_j, q_{j+1}, \xi(j), \xi(j+1)\right] \cdot (\delta q_{j+1} - \delta q_j) - \sum_{j=1}^{n-1} \sum_{h^i|\xi(j)} h^i \frac{\partial V_i}{\partial q}(q_j) \cdot \delta q_j \\ &= T_d'\left[q_{n-2}, q_{n-1}, \xi(n-2), \xi(n-1)\right] \cdot \delta q_{n-1} - T_d'\left[q_0, q_1, \xi(0), \xi(1)\right] \cdot \delta q_0 \\ &\quad - \sum_{h^i|\xi(n-1)} h^i \frac{\partial V^i}{\partial q}(q_{n-1}) \cdot \delta q_{n-1} \\ &\quad + \textstyle\sum_{j=1}^{n-2} \Big(T_d'[q_{j-1}, q_j, \xi(j-1), \xi(j)] - T_d'[q_j, q_{j+1}, \xi(j), \xi(j+1)] - \sum_{h^i|\xi(j)} h^i \frac{\partial V^i}{\partial q}(q_j) \Big) \cdot \delta q_j \\ &= \textstyle\sum_{j=1}^{n-2} \Big(T_d'[q_{j-1}, q_j, \xi(j-1), \xi(j)] - T_d'[q_j, q_{j+1}, \xi(j), \xi(j+1)] - \sum_{h^i|\xi(j)} h^i \frac{\partial V^i}{\partial q}(q_j) \Big) \cdot \delta q_j. \end{aligned}$$

The Euler-Lagrange equations are then

$$\frac{\partial T}{\partial \dot{q}}(\dot{q}_{k+1/2}) - \frac{\partial T}{\partial \dot{q}}(\dot{q}_{k-1/2}) = - \sum_{h^i|\xi(k)} h^i \frac{\partial V^i}{\partial q^i}(q_k), \tag{2.6}$$

If, as is typical, $T_d(\dot{q})$ is quadratic in $\dot{q}$, the system (2.6) gives rise to an explicit numerical integrator that is particularly easy to implement in practice. Lew et al 2003 discuss this algorithm in greater detail.

2.4.1 Multisymplecticity

The right hand side of (2.6) depends on $\xi(k)$, and so the Euler-Lagrange equations for AVIs are time dependent, and do not give rise to a uniform update rule $F(q_{i-1}, q_i) \mapsto (q_i, q_{i+1})$. Instead, consider the total, time-dependent flow $\hat{F}^k(q_0, q_i) \mapsto (q_k, q_{k+1})$. Once again, trajectories satisfying (2.6) are parametrized by $\nu_0 = (q_0, q_1)$, and first variations by $\delta\nu_0 = (\delta q_0, \delta q_1)$. By restricting to such trajectories and variations, the action (2.5) can be rewritten as

$$S_{\text{iAVI}} = \sum_{j=0}^{n-2} \left[\xi(j+1) - \xi(j)\right] T_d\left(\hat{F}^j(\nu_0), \xi(j), \xi(j+1)\right) - \sum_{V^i} \sum_{j=0}^{\lfloor \tau/h^i \rfloor - 1} h^i V_d^i(\hat{F}^{\omega^i(j+1)}(\nu_0)).$$

Then, for $V_d^{i\prime}(q_a, q_b) = \frac{\partial V^i}{\partial q}(q)$,

$$\begin{aligned}
dS_{\text{iAVI}}(\nu)\cdot\delta\nu &= dS_g(\mathbf{q})\cdot\delta\mathbf{q} \\
&= T_d'\left[q_{n-2}, q_{n-1}, \xi(n-2), \xi(n-1)\right]\cdot\delta q_{n-1} - T_d'\left[q_0, q_1, \xi(0), \xi(1)\right]\cdot\delta q_0 \\
&\quad - \sum_{V^i}\sum_{h^i|\xi(n-1)} h^i \frac{\partial V^i}{\partial q^i}(q_{n-1})\cdot\delta q_{n-1} \\
&= T_d'\left[\hat{F}^{n-2}(\nu_0), \xi(n-2), \xi(n-1)\right]\cdot\delta q_{n-1} - T_d'\left[\nu_0, \xi(0), \xi(1)\right]\cdot\delta q_0 \\
&\quad - \sum_{V^i}\sum_{h^i|\xi(n-1)} h^i V_d^{i\prime}\left[\hat{F}^{n-2}(\nu_0)\right]\cdot\delta q_{n-1} \\
&= \theta^-_{\nu_0}\cdot\delta\nu_0 + \theta^+_{\hat{F}^{n-2}\nu_0}\cdot\hat{F}^{n-2}{}_*\delta\nu_0 \\
&= (\theta^- + \hat{F}^{n-2*}\theta^+)_{\nu_0}\cdot\delta\nu_0
\end{aligned}$$

for one-forms θ^- and θ^+. We have

$$0 = d^2 S_{\text{iAVI}} = d\theta^- + \hat{F}^{n-2*} d\theta^+, \tag{2.7}$$

but unlike for synchronous discrete variational integrators, there is no way of relating $d\theta^-$ to $d\theta^+$, and thus discrete symplectic structure preservation is not recovered. Nevertheless, Lew et al 2003 conjecture that this *multisymplectic* structure leads to the good energy behavior observed for AVIs.

2.5 Contact with Discrete Penalty Layers

The above reformulation of AVIs can be leveraged to resolve collisions with guaranteed perfect robustness, and via momentum-symplectic integration, so that the energy behavior of the system as a whole remains good. Consider a standard penalty force approach, which for every two elements A, B and surface thickness η defines the gap function

$$g_\eta(q) = \inf_{a\in A, b\in B} \|a - b\| - 2\eta$$

measuring the proximity of A to B.

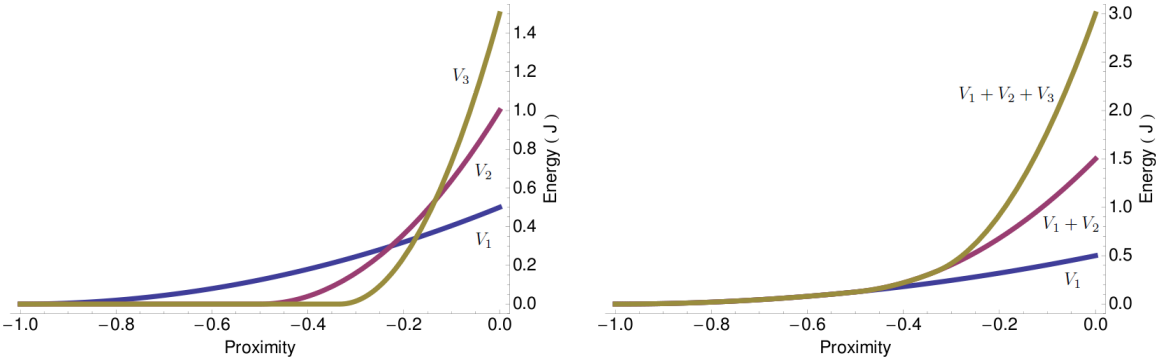


Figure 2.1: Plots of the potential energy of the first three layers as a function of gap function g (left), and a plot of the total potential energy contributed by all layers $\leq n$ for $n = 1, 2, 3$ (right). Notice the potential energy diverges as separation distance approaches 0, guaranteeing that collision response is robust.

The penalty potential is then defined as

$$V(q) = \begin{cases} 0 & g_\eta(q) > 0 \\ k g_\eta(q)^2 & g(q) \leq 0, \end{cases}$$

where k is a user-specified stiffness. As previously discussed, V alone does not robustly prevent interpenetrations: the potential can be viewed as placing a spring between the approaching elements, and for sufficiently large relative momentum in the normal direction, the spring will fully compress, then fail. However, consider placing an infinite family of potentials V_l, $l = 1, 2, \ldots$, between the primitives, where

$$V_l(q) = \begin{cases} 0 & g_{\eta/l}(q) > 0 \\ l^3 k g_{\eta/l}^2 & g_{\eta/l}(q) \leq 0. \end{cases}$$

The region $\frac{\eta}{n+1} \leq d(q) \leq \frac{\eta}{n}$, where exactly n of the potentials are active, is called the n-th *discrete penalty layer*. Figure 2.1 shows a plot of the potential energy of the first few potentials for the case $\eta = k = 1$, as well as the cumulative potential energy of all of the potentials.

The total potential energy of the springs when fully compressed is

$$\sum_{l=1}^{\infty} l^3 k 4 \left(\frac{\eta}{l}\right)^2 = 4k\eta^2 \sum_{l=1}^{\infty} l,$$

which diverges. The infinite array of potentials is guaranteed to stop all collisions. This guarantee in no way depends on the chosen stiffness k: although performance and trajectory will vary with the choice of stiffness, unlike for penalty forces the stiffness does not affect the guarantee. The method is always guaranteed to be robust.

There is one obstruction to implementing this scheme in practice: integrating the l-th spring stably and with good energy behavior requires a time step proportional to $\frac{1}{l^{3/2}}$, which vanishes as $l \to \infty$. Using a traditional integrator, one could decide ahead of time to only simulate the first few springs—but then the guarantee that no penetrations will occur is lost, and the simulation must be run at a prohibitively small time step. AVIs, with the above modifications, and a bit of extra bookkeeping, are a first step towards alleviating the problem, by allowing the user to assign each spring its own time step. The next section describes this bookkeeping.

2.6 The Asynchronous Algorithm

AVIs allow each penalty layer to be assigned a different time step, so that less stiff (l small) layers can take large time steps regardless of the presence of the stiffer layers. However, it is still not possible as a practical matter to integrate the system, since arbitrarily large l would need arbitrarily small time steps, and the global time would never advance. The following observation surmounts this obstacle: at any time during a well-posed simulation, the number of layers that are exerting a non-zero force, or that are *active*, is finite. More precisely, a simulation is well-posed if its total energy over time is bounded—that is, if the simulation begins in a non-penetrating state; all prescribed, infinite-mass bodies are stationary; and only a finite amount of energy is added over time in the form of external forcing. Inactive penalty potentials can be ignored entirely, since they do not change configurational velocity, and the position integration that would take place during the handling of an inactive potential can just as well be done by the following event. Therefore the simulation would be guaranteed to never stop making progress if there is a lower bound for the amount of global time T_g that elapses with the processing of any event. Such a lower bound exists if there is a way to detect which penalty potentials are active or inactive at all times and

remove all inactive events from the priority queue PQ.

Suppose that at the start of the simulation, all penalty layers are inactive. Thus no penalty layer events are needed on the queue. For each pair of simulation elements, the time t_a that the first penalty layer would become active (assuming all elements continue along the trajectory described by their initial velocities) can be calculated, and the corresponding event added to the queue at that time. Such an approach suffers from two problems, however. Firstly, solving for the time when the gap function will be zero is easy in some cases, such as if the elements are two spheres or two planes, but can involve expensive root solves in others, such as if the elements are two non-rigid triangular elements of a thin shell simulation. Secondly, the times computed are fragile: should any event alter the velocity of one of the elements (such as a material force, or gravity, or another penalty force if one of the elements collides with a third party) the activation time is no longer valid and must be recomputed.

Instead of an exact time, only a conservative guarantee, or *certificate* Guibas [1998], that the first penalty layer will not be active before some time t_c (where necessarily $t_c \leq t_a$) is truly needed. For example, one certificate is the existence of an 2η-thick *planar slab* S that separates the two elements up until time t_c, where η is the thickness of the first penalty layer. For an m-dimensional configuration space, such a planar slab is understood to be an extrusion of an $(m-1)$—dimensional affine subspace. Concretely, let w be a unit vector in $\mathbb{R}^m$, w_i be $m-1$ linearly independent vectors in $\mathbb{R}^m$ orthogonal to w, and p a point in $\mathbb{R}^m$. Then the slab $S_{w,p}$ is the set

$$S_{w,p} = \left\{ p + \alpha w + \sum_i \beta_i w_i \middle| -\eta \leq \alpha \leq \eta, \beta_i \in \mathbb{R} \right\}.$$

If such a slab separates the two elements, the first penalty layer cannot become active before t_c. This certificate can be placed as an event on the queue, with time t_c. The certificate might then suffer several fates: Harmon *et al.* [2009]

- An event modifies the velocity of one of the elements before time t_c. The certificate placed on the queue is then no longer valid until time t_c, but instead until a new time t_c' which may be sooner or later than t_c. The algorithm must thus *reschedule* the certificate, by removing its event from the queue, and reinserting it at the appropriate

new time.

- The certificate event is popped from the queue without incident, but it is possible and convenient to find a new separating slab that guarantees the penalty layer does not activate before time $t_c' > t_c$. This new certificate can then be pushed on the queue for time t_c'.

- The certificate event is popped from the queue without incident, but finding a new slab is impossible, costly, or a slab can be found, but the new time t_c' is judged heuristically to be too near t_c. The first penalty layer may then be activated early: doing so affects the efficiency, but not the correctness, of the simulation. Simultaneously, the algorithm searches for an η-thick separating slab to serve as a certificate that layer two is not yet active, and the whole process described above is repeated.

Detecting when a penalty layer event becomes inactive, and should be removed from the queue, is much simpler than detecting layer activation: whenever a penalty force for layer n is integrated, the algorithm simply checks if the force applied was 0. If so, and if the two elements in question are separating, layer n is now inactive: it is not pushed back onto the queue (and instead a separating slab of thickness η/n is sought.)

It is very important to note that when an event becomes active and is added back into the event priority queue, it is done so at a time that is *an integer multiple of its timestep from its last time of integration.* That is, those times when integration would do nothing have been optimized away, but the potential's "integration clock" has not been tampered with or realigned, since every potential having a fixed-size time step was fundamental to the proof that asynchronous variational integration is multisymplectic. The spring-on-a-plane example described below underlines the danger of failing to maintain such a fixed time step.

For an event E, denote all simulation elements on which E depends the *support* of V. Denote all simulation elements whose velocities are modified by E the *stencil* of E. For force integration events, there is no distinction between stencil and support. Certificates have a support, but no stencil. Algorithm 1 uses this terminology to incorporate the above into the AVI algorithm.

In Algorithm 1 and its accompanying sub-algorithms, the behavior of the functions

Algorithm 1 Proposed algorithm for asynchronous contact resolution.

Let force events be (potential, time step, time) triplets $E = (V, h, t)$.

Let PQ be a priority queue of events, sorted by event times $E.t$.

$T_g \leftarrow 0$ ▷ T_g maintains the value of the simulation clock

$q \leftarrow q_0$ ▷ Set up initial conditions

$\dot{q} \leftarrow \dot{q}_0$

Push non-penalty (e.g. material) events on the queue

for all pairs of elements K_1, K_2 **do**

    $E \leftarrow$ FindCertificate(K_1, K_2)

    PQ.push(E)

end for

loop

    $E \leftarrow PQ$.pop

    $q \leftarrow q + (E.t - T_g)\dot{q}$

    if E is a force event **then**

        handleForceEvent(PQ, E)

    else

        handleCertificateEvent(PQ, E)

    end if

    $T_g \leftarrow E.t$ ▷ Update the simulation clock

end loop

Algorithm 2 handleForceEvent

Require: Priority queue of events PQ and force event E that needs processing

▷ Processing a force event E is a three-step process: integrating the force, rescheduling all events whose support depends on E's stencil, and lastly, resceduling E itself.

for all i in Stencil(E) **do**

$\dot{q}_i \leftarrow \dot{q}_i - (E.h)M_i^{-1}\frac{\partial E.V}{\partial q_i}$ ▷ Update only those elements affected by this event.

end for

▷ Reschedule all events whose support depends on E's stencil

for all Certificate events E' with Stencil(E) $\cap$ Support(E') $\neq \emptyset$ **do**

PQ.remove(E')

Schedule(E')

PQ.push(E')

end for

▷ If E was a penalty force event, it exerted 0 force, and the two primitives in question are separating, then we no longer need it

if E is a penalty force event and $\frac{\partial E.V}{\partial q_i} = 0$ **then**

if $E.V.K1$ and $E.V.K2$ have positive relative velocity (are separating) **then return**

end if

if addCertificate($E.V.K1, E.V.K2$) **then return**

end if

end if

▷ Otherwise, reschedule E itself

PQ.push($V, h, t + h$)

Algorithm 3 handleCertificateEvent

Require: Priority queue of events PQ and certificate event E that needs rescheduling

if not addCertificate($E.K1$, $E.K2$) **then**

▷ Finding a new certificate failed. We must thus activate a penalty force, one layer deeper than the deepest currently active penalty force event.

$CurLayer \leftarrow \max_{\{\text{penalty events } E' \text{ on queue for } E.K1 \text{ and } E.K2\}} E'.layer$

$E' \leftarrow$ new PenaltyForceEvent($E.K1$, $E.K2$, $CurLayer + 1$)

PQ.push(E') ▷ Push the appropriate penalty force event on the queue

end if

Algorithm 4 addCertificate

Require: Priority queue of events PQ, and two elements $K1$ and $K2$

▷ Attempts to find a certificate for the collision of $K1$ against $K2$ and add it to the queue. Returns true if one was found.

$E' \leftarrow$ FindCertificate($K1$, $K2$)

if E' was successfully found **then**

PQ.push(FindCertificate($K1$, $K2$)) **return** true

end ifreturn false

FindCertificate and Schedule will depend on the type of certificate chosen. FindCertificate returns a new certificate for a given pair of elements, if possible and practical, and Schedule computes the time a certificate becomes invalid, as described in the paragraphs above. For thin shell simulation, where all simulation elements are convex triangles, edges, and vertices, separating slabs serve as ideal certificates, since it is cheap to compute Schedule, in this case by calculating element-plane intersection times. Although any choice of certificate, and heuristic for when to abort searching for a new certificate, preserves the correctness of the algorithm, the progress property described in the first paragraph of this section relies on the certificates efficiently weeding out inactive events so that some certificate is found before all (infinitely many) layers for a pair of elements are activated. No problems have been observed using separating slabs for thin-shell simulations, but different certificates may be needed, e.g., for concave rigid bodies.

2.6.1 Further Optimizations

The technique explored in the previous section, of finding a sequence of conservative certificates guaranteeing that some property holds, instead of calculating an exact time when that property stops holding, is the central idea behind a wide class of algorithms known as *Kinetic Data Structures* (KDSs) Guibas [1998]. In the case described above, the property was inactivity of a given penalty layer. KDSs are particularly well-suited for an asynchronous approach, since certificate expiration times may not all align to some convenient simulation clock, and the required rescheduling of certificates/searching for new certificates can reuse the priority queue data structure already needed for force integration events. To improve the efficiency of the implementation used to create the examples below, several more KDSs in addition to the separating slabs discussed above were implemented: a bounding volume hierarchy Klosowski *et al.* [1998] was used to take advantage of the fact that spatially distant elements are unlikely to collide, separation lists Weller and Zachmann [2006] to optimize the bookkeeping of this hierarchy, and a novel KDS was devised to leverage the observation that high-frequency, low amplitude oscillations in velocity do not significantly change a separation slab's expiration time, so that rescheduling is in many cases unnecessary. All of the improvements are described in greater detail in Harmon *et al.* [2009].

2.7 Dissipation

The Coulomb friction model is a simple approximation to kinetic friction: at a point of contact between two bodies, the Coulomb force has magnitude $\mu|F_n|$, where μ is a coefficient of friction and F_n is the normal force at the contact points, and has direction opposite the relative tangential motion of the contact points.

Whenever an impulse is applied during integration of a penalty layer, a corresponding frictional impulse can also be applied. Just as increasingly stiff penalty forces are applied for contact forces, friction forces are increasingly applied (equal to $\mu|F_n|$) to correctly halt high-speed tangential motion. Notice that these friction forces, like the material and contact penalty forces, are applied asynchronously: every layer applies friction independently at its own time step.

This simple, asynchronous formulation of friction fits very naturally into the framework of AVIs. Unfortunately, it is unsuitable for simulations featuring static friction, such as a block of wood resting on an inclined plane. The above formulation, with friction applied piecemeal during penalty integration, is reactive instead of proactive, and in simulations of this type the block of wood has been observed "creeping" down the incline no matter how high a coefficient of restitution is chosen. A more comprehensive model of friction compatible with the AVI framework, which correctly handles static friction, remains future work.

2.8 Results and Discussion

By resolving contact using nested penalty layers, our method guarantees that no collisions occur during simulations, without compromising the multisymplectic structure, and hence the good energy behavior, of AVIs. A full set of numerical experiments demonstrating these properties are described by Vouga et al 2011; two representative examples are reproduced here.

Sphere-plate Impact The impact experiment of a hollow spherical thin shell against a thin plate, as described in Cirak and West's article on Decomposition Contact Response

(DCR) Cirak and West [2005], was reproduced using the proposed framework. A sphere of radius 12.5 cm approaches a plate of radius 35 cm with relative velocity 100 m/s. Both the sphere and the plate have thickness 0.35 cm. The time steps of the material forces (stretching and bending) are 10^{-7} s (the same as those chosen by Cirak and West.)

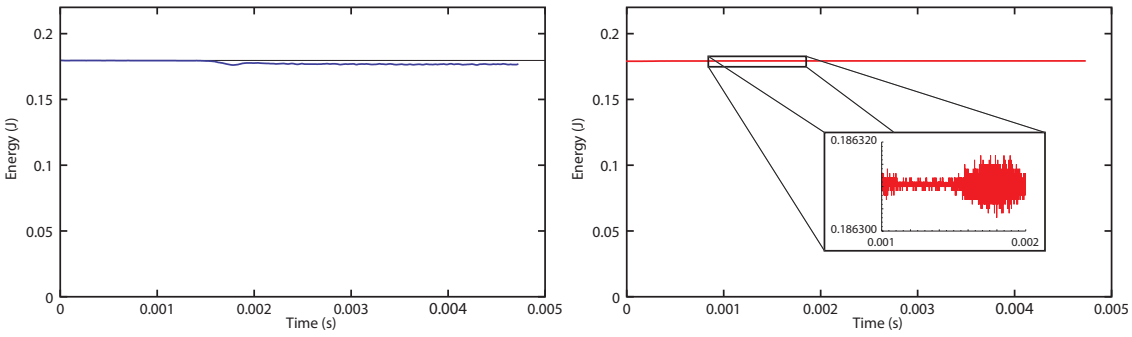


Figure 2.2: Total energy over time of a thin sphere colliding against a thin plate, simulated using the proposed contact response method (right) compared to data provided for decomposition contact response Cirak and West [2005] (left).

Figure 2.2 compares energy over time when this simulation is run using both the proposed method and DCR. Using the former there is no noticeable long-term drift; closely examining the energy data reveals the high-frequency, low-amplitude, qualitatively-negligible oscillations characteristic of symplectic integration. The latter introduces noticeable artifical energy decay.

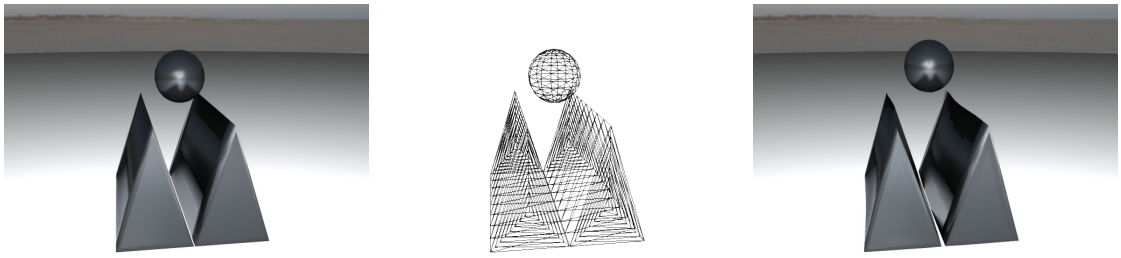


Figure 2.3: A sphere falling into a wedge, at the beginning of the simulation (left and center) and 0.42 seconds later, after the sphere has reflected off of the wedge (right). The center figure shows the mesh elements of the bodies.

Wedge Collision Inspired by Pandolfi et al 2002, a rigid thin-shell sphere was dropped into a wedged formed by two thin shell triangular prism, shown in Figure 2.3. Each prism has an isosceles base with width 12.92 cm and height 20.05 cm, and length 38.41 cm. The prisms contain 71 vertices each. The sphere contains 92 vertices, has radius 4.97 cm and begins the simulation 20.84 cm above the ground plane on which the prisms rest. The sphere has initial downwards velocity of -100 cm/s (no gravity). The sphere and shells use the same thin shell model as the debris in the above trash compactor example, with bending and stretching stiffness parameters 100000 and 50000 respectively.

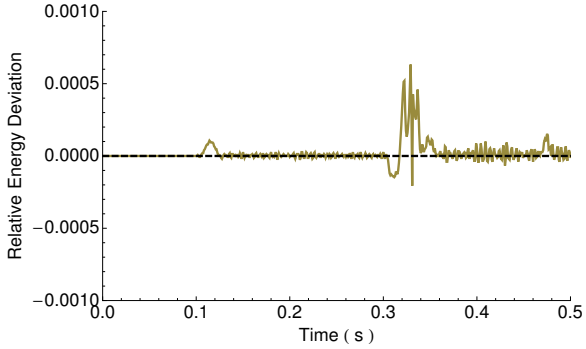


Figure 2.4: The relative error in energy of the wedge-sphere system as a function of time. The energy oscillates about its initial value without drift.

As the sphere descends, it enters into multiple contact with the faces of the wedge, which undergo elastic deformation and high-frequency vibration. Despite the large areas of simultaneous contact and high velocity at the time of impact, the energy of this system, plotted in Figure 2.4, exhibits good behavior and does not drift.

Discussion The above asynchronous, multisymplectic approach to simulating contact offers unique guarantees of safety against collisions without compromising good momentum and energy behavior. There remain, however, several exciting directions for long-term future work:

- As discussed above, Vouga et al 2011 describes how to incorporate several kinds of dissipation into the above framework, including the coefficient of restitution, and kinetic friction. However, it is not yet known how to incorporate static friction. Static

friction conflicts fundamentally with asynchrony: in an asynchronous simulation, contact between a pair of elements is resolved piecemeal, by summing the impulses at many different times contributed by many different penalty layers. At any given moment of time it is unclear how to define a total normal force, an element necessary for the robust treatment of even the most elementary static friction models. Successfully merging the handling of friction with the asynchronous framework, to allow simulations of systems such as a standing house of cards, remains future work.

- The precise algorithms used for detecting *which layers* are active and need to be integrated at any given time was is immaterial to the method's guarantees. The choice does dramatically effect the method's efficiency, however; therefore there remains much room for further work on *conservative optimizations* that improve efficiency without violating the established guarantees. The next chapter discusses some of my newer collaborative work Ainsley *et al.* [2012] in this direction based on speculative parallel simulation, which is over 200 times faster than the original Harmon *et al.* [2009] approach using kinetic data structures Guibas [1998] summarized above.

- Nested penalty layers could be generalized to arbitrary potential functions that "activate" in response to entering some region of configuration space. Perhaps this idea could be used to incorporate *spatial adaptivity* into the variational framework, unifying this work on contact with that on developable surfaces discussed in Chapter 5.

Chapter 3

Improving Performance of the Asynchronous Contact Mechanics Algorithm with Speculation

The algorithm described in Chapter 2, by discretizing a barrier potential as infinitely many nested quadratic potentials ("layers") and then integrating these potentials using an asynchronous variational integrator, possesses unprecedented robustness and correctness guarantees. These guarantees make the ACM contact response framework ideal for simulating thin objects like shells and cloth undergoing contact, where interpenetrations are otherwise easy to introduce, difficult to detect, and fatal to the output of the simulation.

Unfortunately, the guarantees come at a performance price, and the KDS implementation of Harmon et al 2009 summarized in Chapter 2 (the *original* or *KDS implementation*) is for typical, large-scale cloth simulations up to three orders of magnitude slower than other state-of-the-art algorithms Bridson *et al.* [2002]; Selle *et al.* [2009] that do not have the guarantees. In a follow-up collaboration Ainsley *et al.* [2012], I developed a new implementation of ACM that dramatically decreases the amount of time spent on bookkeeping and collision detection, speeding up the original implementation by over an order of magnitude. Moreover, while the original ACM was difficult to parallelize, by employing *speculation* to expose easy parallelization, we arrive at another order of magnitude speedup. The new implemen-

tation, described below, therefore yields speedups of more than two orders of magnitudes on a 12-core work station, enabling practical simulations of complex contact geometries.

3.1 Overview

An alternative to KDSs The original approach to activating penalty layers guarantees that collisions cannot be missed. Unfortunately, this guarantee carries a heavy cost: in typical simulations over 90% of the execution time is spent on rescheduling or processing KDS events Harmon [2010]. Moreover, every time an event is processed, the event queue changes in a way that is unpredictable a priori: a KDS event will either reschedule itself for an unknown time in the future, or will insert a new penalty layer whose first tick could occur an arbitrarily small amount of time later and will itself cause rescheduling of other KDS events. Because of these unpredictable causal dependencies between events, processing and rescheduling of KDS events cannot be easily parallelized.

Instead of using KDSs to guarantee that the simulation never enters an interpenetrating state, we propose taking advantage of the *time warp* paradigm Jefferson [1985]: we simulate a window of time without attempting to find or resolve any new collisions. At the end of the window we perform retrospective collision detection, and if any collisions were missed, we *roll back* to the beginning of the window, add new penalty layers, and repeat this process until no collisions are detected. We describe this algorithm in detail in Section 3.3.

Speculative simulation and rollback is not at first glance an obviously fruitful model for collision detection and response. Consider, as a point of comparison, a hypothetical *ideal* implementation of ACM that wastes no work on collision detection: an omniscient oracle informs this ideal implementation exactly when to activate all penalty layers needed to resolve imminent collisions, and when to deactivate the penalty layer because it will exert zero force the next time it is processed. This ideal implementation gives a lower bound on the amount of computation *any* functionally identical optimization of ACM must perform.

It is useful to compare this ideal implementation to both the original ACM implementation and the proposed rollback scheme: any work done by either method beyond that of the ideal implementation is termed *wasted* work. A rollback scheme performs several types

of wasted work:

- **Resimulation**: Every time an event is processed and then rolled back, the time spent processing that event was wasted.
- **Collision detection**: Time spent on collision detection takes away time that could have been spent on integrating forces and advancing the simulation. In the ideal implementation, collision detection is instantaneous.
- **Unneeded penalty forces**: Penalty events that exert zero force (as a result of being added to the event queue by overly-conservative collision detection) do not need to be processed.
- **Bookkeeping**: Rollback has significant miscellaneous overhead: saving and restoring the (large amount of) simulation state after every rollback window, gathering the trajectory data during each window needed by collision detection, and so on.

In the original implementation, processing and rescheduling of KDS events are the main sources wasted work, in addition to event queue maintenance. (Since the event queue contains many KDS events in addition to force events, pushing, popping, and rescheduling events on the queue is more costly.)

Despite the potential overhead of speculative simulation with rollback, this approach has several advantages over original ACM: the collision detection at the end of each simulation window needs only to determine *whether or not* a collision has occurred (the time of impact is unimportant), and can take advantage of four instead of three dimensions of information (see Section 3.4). As a result, collision detection and resimulation is substantially cheaper than rescheduling and processing KDS events "along the way." Together these factors significantly reduce the amount of wasted work incurred during a typical simulation. Moreover, both the collision detection and the processing of material/penalty forces within each window can be easily parallelized (Section 3.5), unlike rescheduling of KDS events.

3.2 Contributions

The algorithms described in this chapter take the only contact response algorithm with guaranteed safety, progress, and correctness (see chapter 2) and improve its efficiency by over two orders of magnitude, without compromising any of the original guarantees. Although the Time Warp paradigm has been successfully used before for contact response, especially for rigid body simulation (see section 1.2.4 for a discussion of recent approaches), we show that it is especially well-suited for our asynchronous algorithm. Time Warp allows us to take advantage of a novel divide-and-conquer narrow phase collision detection algorithm (section 3.5) to more efficiently detect changes in the active penalty layers. Moreover, as described above, whereas the original ACM algorithm did not expose any obvious opportunities for profitable parallelization, we show how to take advantage of the embarrassingly parallel structure of the speculative reformulation of ACM for an additional order of magnitude speed improvement (section 3.5).

3.3 Speculative Simulation with Rollback

In place of forward-looking kinetic data structures, we propose detecting collisions in hindsight and resolving them with a speculative model in the spirit of Jefferson's *time warp* algorithm 1985. We tile time into consecutive fixed-sized *rollback windows* of duration R. (We discuss the choice of the parameter R in Section 3.6.1.) Within each window, we advance the simulation by processing events as described in the original ACM algorithm, except that we do not add, process, or reschedule any KDS events. That is, we process internal force events, and penalty events for any penalty layers that were active at the start of the window. More collisions may occur during the rollback window but we make no attempt at detecting or responding to them before the end of the rollback window.

We stop processing events when we would process the first event whose time exceeds the end of the rollback window. We then perform interference detection to determine whether any collisions were missed during the window: if so, we restore the entire simulation to its state at the beginning of the rollback window, and activate a penalty layer for each missed collision. We then resimulate the rollback window. The forces from the new penalty layers

Algorithm 5 Algorithm to integrate one window

1: **procedure** IntegrateWindow
2: $\mathrm{X}_0 \leftarrow \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots\}$ ▷ save start-of-window positions
3: $\dot{\mathrm{X}}_0 \leftarrow \{\dot{\mathbf{x}}_0, \dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2, \ldots\}$ ▷ save start-of-window velocities
4: $Q_0 \leftarrow Q$ ▷ save start-of-window queue state
5: **repeat**
6: H $\leftarrow \mathrm{X}_0$ ▷ reset histories
7: **while** Q.**head**.$t < t_0 + R$ **do**
8: $(E, h, t) \leftarrow Q$.**pop** ▷ Pop event E with time step h at time t
9: ProcessEvent(E,h,t)
10: **end while**
11: $\mathcal{C} \leftarrow$ BroadPhaseCollisionDet(H) ▷ get missed collisions
12: **if** $\mathcal{C} \neq \emptyset$ **then**
13: $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots\} \leftarrow \mathrm{X}_0$ ▷ restore positions
14: $\{\dot{\mathbf{x}}_0, \dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2, \ldots\} \leftarrow \dot{\mathrm{X}}_0$ ▷ restore velocities
15: $Q \leftarrow Q_0$ ▷ restore start-of-window queue state
16: Q.**push**(new penalty events constructed from C)
17: **end if**
18: **until** $\mathcal{C} = \emptyset$
19: **end procedure**

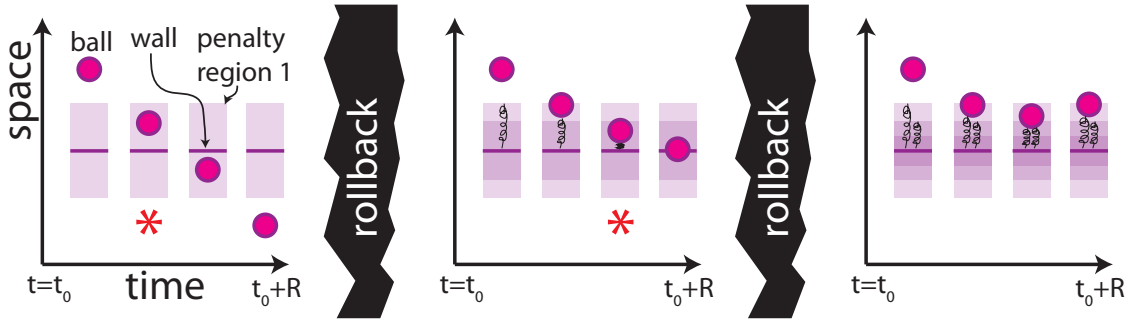


Figure 3.1: A cartoon illustrating the rollback process: the simulation steps a ball forward in time, heedless of collisions *(left)*. Interference detection at the end of the rollback window notices that the ball enters the first penalty layer, so the simulation rolls back, activates the first penalty layer, and resimulates the window *(middle)*. During resimulation, the ball enters the second penalty layer; rolling back, activating the second penalty layer, and resimulating, no collisions (i.e., entries into third penalty layer) are detected *(right)*, and the result is finally accepted.

may have induced additional collisions, or may not have been sufficient to stop the detected collisions, so we repeat this process until collision detection reports no missed collisions during the rollback window. See Figure 3.1 for an illustration of this process.

Implementation At the start of every rollback window we take a snapshot of the entire simulation state, which we restore in the event of a rollback. This includes positions, velocities, events on the queue and their times, positions of the material reference configuration, etc. We also maintain a *history* of vertex positions: for each vertex, we track its position at the start of the window, the end of the window, and its position and time whenever it changes velocity due to force processing. Since vertices move along piecewise linear trajectories, this history minimally encodes the entire trajectory of the simulation over the rollback window, and is passed to the collision detection algorithm at the end of the window. Note that due to ACM's asynchrony, different vertices have different numbers of history entries and entries do not necessarily align in time.

At the end of every rollback window, if a penalty layer is exerting zero force, we remove it from the list of active penalty layers.

Algorithm 6 Algorithm to process one event

1: *Process an event E with time step h, scheduled time t*
2: **procedure** PROCESSEVENT(E,h,t)
3: $\xi := \textbf{stencil}(E)$ ▷ global indices of the local stencil
4: **for** $i \in \xi$ **do** ▷ update positions and clocks
5: $\mathbf{x}_i \leftarrow \mathbf{x}_i + (t - t_i)\dot{\mathbf{x}}_i$
6: $t_i \leftarrow t$
7: **end for**
8: compute $\mathbf{F}_\xi$ ▷ local impulses $\mathbf{F}_i$ for $i \in \xi$ (embarrassingly parallel)
9: $\dot{\mathbf{x}}_\xi \leftarrow \dot{\mathbf{x}}_\xi - h\mathrm{M}_\xi^{-1}\mathbf{F}_\xi$ ▷ update velocities (embarrassingly parallel)
10: $Q.\textbf{push}(E, h, t + h)$ ▷ Schedule recurring event
11: **for** $i \in \xi$ **do** ▷ save history
12: $\mathrm{H}_i.\textbf{append}(\mathbf{x}_i, t)$ ▷ H_i records trajectory of $\mathbf{x}_i$
13: **end for**
14: **end procedure**

3.4 Collision Detection

At the end of every rollback window, we must examine the trajectory of the system over the course of the window and determine if any collisions occurred. In particular, each pair of material primitives (edge-edge or vertex-face) either has no active penalty layers—in which case we must detect if we need to activate the first, outermost penalty layer for that pair—or they already have some penalty layers on the queue, in which case we must check if the next-deeper layer is needed. To preserve the *safety* guarantee we must perform continuous-time proximity detection on the full trajectory: it is not enough to merely check positions at the end of the window as objects may have tunneled through each other. This proximity detection is the same as collision detection with an offset surface corresponding to the thickness of the penalty layer.

Our particular problem domain has several distinguishing features:

- Each vertex in the simulation moves in a piecewise linear trajectory, and vertices do not change trajectory in lockstep.

- Although the coarse motion of a vertex over the rollback window might be simple, for simulations involving cloth, thin shells, or other objects with stiff internal forces, the fine trajectory is composed of very many high-frequency, low-amplitude oscillations.
- The interference distance we need to detect against differs for each pair of primitives in the simulation, since we always check for that pair entering its next deeper penalty layer, and different primitive pairs in the simulation have different numbers of penalty layers already active.
- We only need to know whether or not a collision occurred at some point during the rollback window – the precise time of impact is unimportant.

We propose a three-phase detection algorithm with these features in mind.

Broad phase: swept-volume k-DOPs We cull collisions between primitives that remain spatially distant for the entire rollback window by fitting a k-DOP hierarchy Konečný and Zikan [1997] to the swept volumes of the triangles in the simulation. We have observed that (unoriented) 26-DOPs work well in practice. We place bounding volumes around swept volumes rather than around each triangle at each point in time to avoid a complete rebuild of the hierarchy at the end of each rollback window. We also avoid scenarios in which increasing history granularity leads to large hierarchies and therein costly traversal.

Since we need to detect proximity between pairs of primitives, and since that proximity varies depending on how many penalty layers are already active for that pair, during fitting we inflate the k-DOPs by the conservative, largest penalty layer thickness η_1 (the thickness of the first penalty layer). If two leaf nodes (triangles) overlap, we look at all possible pairs of primitives taken from the two triangles, look up how many penalty layers k (if any) are already active for that pair, and check if their swept volume k-DOPs, inflated by η_{k+1}, overlap. If so, we proceed to the narrow phase.

Narrow phase: Space-time separating planes Given an edge-edge or vertex-face pair that could not be culled by the broad phase, the narrow phase must determine if the primitives come within some proximity η_{k+1}. Each of the vertices that compose the pair move in a piecewise linear trajectory, so the rollback window can be subdivided into time

intervals during which all vertices have constant velocity; in this setting, proximity detection amounts to finding the roots of well-known continuous collision detection polynomials (with thickness) of degree at most six Stam [2009]; Brochu *et al.* [2012]. Such root solves are very expensive, so we propose a second phase of culling based on separating slabs that takes advantage of the fact that we do not need the time of impact, and that over a small window of time most forces only cause small perturbations to positions.

Algorithm 7 Algorithm for culling narrow-phase collisions on the interval $[t_0, t_f]$

```
1:  procedure NarrowPhase(t_0, t_f)
2:      if (t_f ≤ t_0) then
3:          return false
4:      else if (t_f − t_0 < tol)∧ ConstantVelocity(t_0, t_f) then
5:          return ContinuousCollisionDetection(t_0, t_f)          ▷ Fail-safe
6:      end if
7:      t_mid = (t_f + t_0)/2
8:      if PrimitivesProximate( t_mid ) then
9:          return true
10:     end if
11:     (t_l, t_u) = NoCollisionInterval( t_mid )
12:     return NarrowPhase(t_0, t_l) ∨ NarrowPhase(t_u, t_f)
13: end procedure
```

Algorithm 7 outlines our approach. For a rollback window beginning at time t_0 and ending at $t_f = t_0 + R$, we first calculate the distance between the primitive pair at the midpoint $t_{\text{mid}} = \frac{t_0+t_f}{2}$ (line 8). If the pair is within proximity at this time, we know a collision must occur during the rollback window. Otherwise, a separating slab of thickness η_{k+1} must exist and certify the lack of collisions on some interval (t_l, t_u) around t_{mid} (line 11). If $t_l < t_0$ and $t_u > t_f$, we are guaranteed that the pair does not collide for the entire rollback window. If $t_0 < t_l$ we recursively apply this algorithm to the time window $[t_0, t_l]$, and similarly for $[t_u, t_0 + R]$. Figure 3.2 illustrates this algorithm.

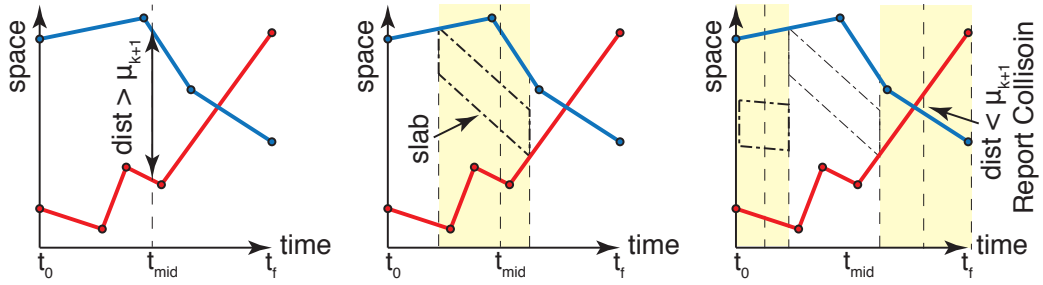


Figure 3.2: Narrow phase collision detection using separating slabs. The trajectories of two vertices in 1D are shown. At t_{mid} we detect that the two vertices are sufficiently far apart, so cull an interval of time based on how long a separating slab can be shown to certify that there are no collisions. We then recurse until we have found a collision or proven that no collisions occur at any time in the rollback window.

Failsafe: Continuous collision detection If we would recurse on an interval that is too small (we use 10^{-10} seconds for this tolerance) and the primitive pair has constant velocity within this interval, we invoke the third phase of collision detection, CTCD using root solving (culling some polynomials when we can prove using interval arithmetic that they have no roots during the rollback window), as a last resort (line 5). Invoking the failsafe is rare: in our benchmark examples, only about 0.2% of all narrow phase calls require the failsafe.

3.5 Parallelization

In the original ACM code, after every event is processed, KDS events associated to the vertices in the event's stencil must be rescheduled. This rescheduling alters the event queue in unpredictable ways: the KDS event might reschedule itself for an arbitrary later time, or it might remove itself from the queue and activate a penalty layer; it is therefore unclear how this rescheduling could be effectively parallelized. On the other hand, removing KDS events as the mechanism for guaranteeing collision-free simulations, and replacing them with speculative simulation followed by collision detection, opens the door to straightforward parallelization of the entire algorithm.

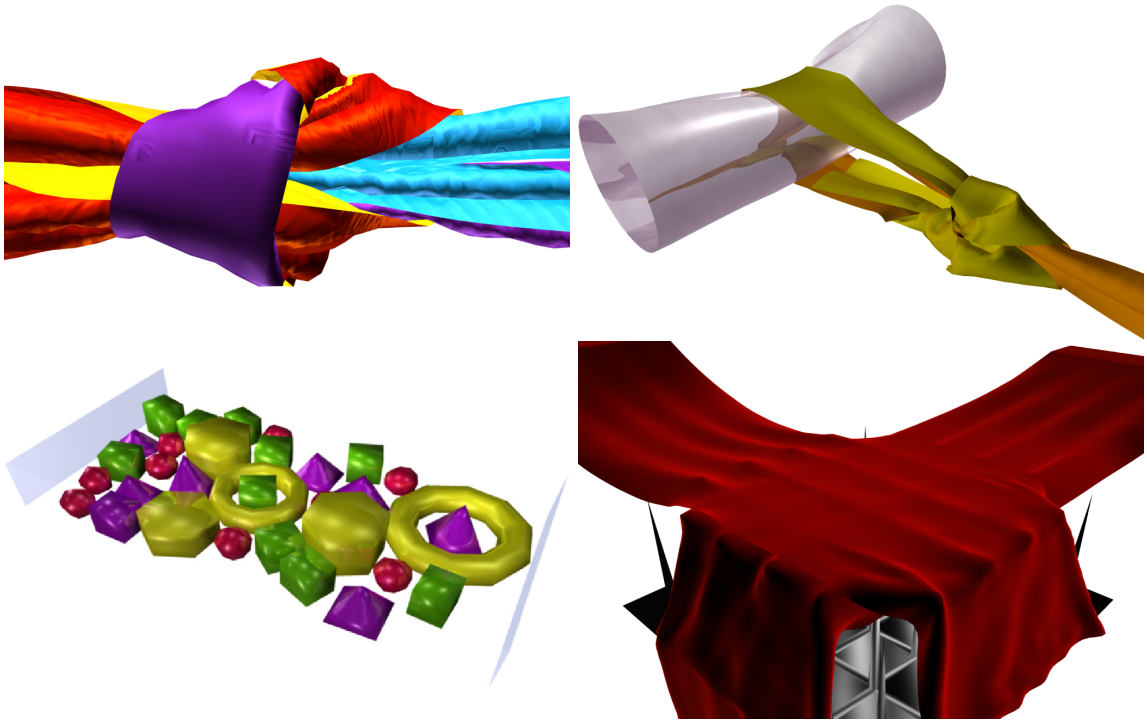


Figure 3.3: The benchmarks from Harmon et al 2009. From left to right: reef knot, bowline knot, trash compactor, and two cloths draped. Timing comparisons for these benchmarks are listed in Table 3.1.

3.5.1 Parallelizing Force Processing

As in the ACM paper, we are concerned primarily with simulating meshes of approximately uniform resolution; for such meshes, the maximum stable time step for the internal forces does not vary much, and so as in the original ACM implementation we *conservatively bucket* material forces into super-elements Huang *et al.* [2007]; Harmon [2010] by setting all of their time steps to that of the stiffest element. For example, gravity forces, internal forces, and the different layers of penalty forces are each grouped into their own bucket. We then represent the group as a single item on the event queue. Once bucketed, internal forces can be parallelized very simply without synchronization: to process the grouped forces we integrate positions to the current time (see Alg. 6 line 4), compute and store for each force in the group the impulse applied by that force (see Alg. 6 line 8), and iterate over the vertices of the simulation, applying to each velocity the sum of the impulses computed in the previous step (see Alg. 6 line 9).

Each of these steps is done in parallel. Penalty layers of the same depth can be naturally grouped since they have identical stable timesteps; we process them in parallel in the same way.

For simulations involving graduated meshes of widely varying triangle size, bucketing coarse element at the fine elements's time step is inefficient. An interesting direction for future work would be to explore using several buckets at different orders of time step magnitude to better handle such a distribution of internal force stiffnesses.

3.5.2 Parallelizing Collision Detection

To get reasonable scaling behavior, collision detection must also be parallelized. It is trivial to run the narrow phase (both the spacetime separating slabs and the root solve) in parallel. Parallelizing the broad phase effectively is more complicated, so at present we opt for a simple staggering scheme that allows us to run a sequential broad phase while still making use of all available cores by allowing the simulation to proceed to the next rollback window in parallel. A number of better methods have been proposed for efficient parallel collision detection Kim *et al.* [2009]; Pabst *et al.* [2010]; Tang *et al.* [2010, 2011], and we hope to incorporate this into our framework in the future.

In our current implementation, whenever we need to perform collision detection, we run it in parallel with optimistic simulation of the next rollback window. In other words, after simulating rollback window i, we perform collision detection on window i, and with any remaining cores begin simulation of window $i + 1$ under the assumption that no collisions will be found. If collision detection does return a collision, we immediately stop simulating frame $i + 1$ and roll back to the beginning of frame i. Figure 3.4 illustrates this timeline. Speculatively starting integration of the next frame is advantageous whenever collision detection ultimately finds no collisions during the previous window – in our benchmarks, this occurs 60–70% of the time.

3.5.3 Miscellaneous Optimizations

Several other optimizations we attempted further improved the performance of our framework.

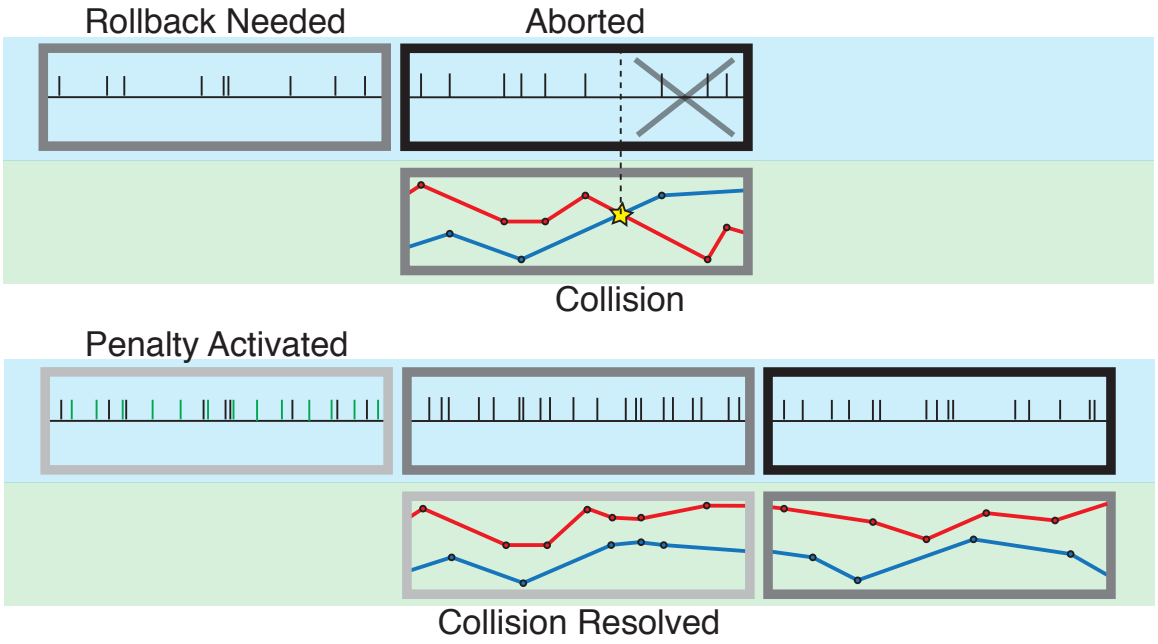


Figure 3.4: After simulating a rollback window (*top-left*), we optimistically continue simulating the next window concurrently while performing collision detection (*top-middle*). If a collision is detected, we interrupt the simulation and roll back (*bottom-left*). If collision detection confirms there were no collisions in the last window, we continue simulating (*bottom-right*).

- **Kernel fusion of the bending and stretching forces**: for simulations involving both bucketed bending and stretching membrane forces, we compute the force contribution of the stretching force at the same time as that of the bending force. Doing so halves the number of times mesh position information must be fetched and improves cache performance.

- **Bandwidth reduction**: Using reverse Cuthill-McKee reordering Cuthill and McKee [1969], we reduce the number of cache misses incurred while integrating internal forces by reordering the mesh vertices at the start of the simulation. Other methods for cache aware or cache oblivious layouts may be even more effective Yoon *et al.* [2005].

In addition to the changes listed above, we also refactored and micro-optimized the code in several places (for instance, removing unnecessary trigonometric function calls in the

bending force computation, unrolling tight inner loops, and rearranging the layout of data structures in memory to maximize cache efficiency). These changes already improved the performance of our code when run with a single thread. However, more importantly, we found that such low-level optimizations aimed at improving cache performance were essential to achieving reasonable scaling behavior with increasing number of cores.

3.6 Analysis and Results

Example	Original ACM	One thread, no slabs	One thread	Twelve threads	**Total speedup**
Reef Knot	23.6 hrs	74 mins	50.5 mins	5.8 mins	**244x**
Bowline Knot	7.0 days	4.9 hrs	121 mins	15.2 mins	**663x**
Trash Compactor	13.8 hrs	53.5 mins	13.3 mins	2.4 mins	**345x**
Two Cloth Drape	11.6 days	6.7 hrs	4.7 hrs	40.1 mins	**416x**

Table 3.1: Wall clock time for each of the examples benchmarked by Harmon et al 2009. We ran each examples using the publicly-available ACM implementation, our code with only a single thread and CTCD only (instead of spacetime separating slabs) for the collision detection narrow phase, our complete code with only a single thread, and our complete code with 12 threads. The simulation parameters were identical to those selected by Harmon et al. For the rollback window size R we used 1/300 for all examples.

Existing benchmarks We ran our method on four of five examples timed by Harmon et al 2009; see Figure 3.3. We used an Intel 12-core Westmere-EP workstation (X5690 @ 3.47GHz). To ensure a fair comparison we also re-timed the publicly released ACM code on identical hardware. Table 3.1 shows the timing comparison; our method is more than two orders of magnitude faster than the original ACM code for all examples. Due to needing fewer events on the queue, our implementation is also more memory efficient – for the Reef Knot, our implementation uses 229 MB of RAM, compared to 424 MB for the original implementation.

We stress that this method is a conservative optimization of the original, KDS-based asynchronous algorithm: we propose changing *how* collision detection is performed (an implementation detail), but not *which* collisions are detected or how these collisions are

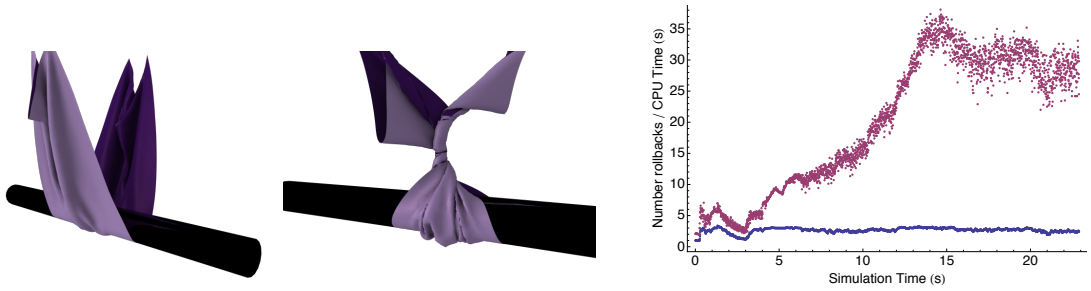


Figure 3.5: A spinning rod wrings out a sheet of cloth. Our method's safety guarantee, which we inherit from ACM, ensures that no interpenetrations occur even as the cloth twists tightly around itself multiple times. We plot the number of times each frame rolls back (*blue*) and the total wall clock time spent computing each frame (*maroon*). As a point of context, the rod starts spinning at time 3.0 and stops spinning at 12.0.

resolved. In particular, our implementation preserves the three guarantees of safety, correctness, and progress.

The twister Inspired by the video accompanying Bridson's thesis 2003, we simulated a cylindrical rod suspended within a cloth cradle, Figure 3.5. Constant external torque applied to the rod wrings out the cloth. Since our method preserves all of ACM's guarantees, including *safety*, no interpenetrations occur over the course of the simulation, despite the amount of self-contact.

Scaling of parallelism We ran benchmark examples on an Intel 32-core Westmere-EX machine (E7-8837 @ 2.67GHz), and varied the number of cores our code was permitted to use. Figure 3.6 shows how the wall-clock time varies as a function of cores used. We observe reasonable scaling for up to 16 cores, with additional cores providing diminishing returns.

Profiling our code suggests to us that cache performance during event processing limits our scalability. When integrating bucketed force events, we first compute and store the force supplied by each force event, then iterate over the vertices and for each vertex look up and apply each relevant force's impulse contribution to that vertex's velocity. Splitting integration into these two steps allows us to parallelize both steps without synchronization, but increases each force event's cache footprint by requiring the impulses to be temporarily

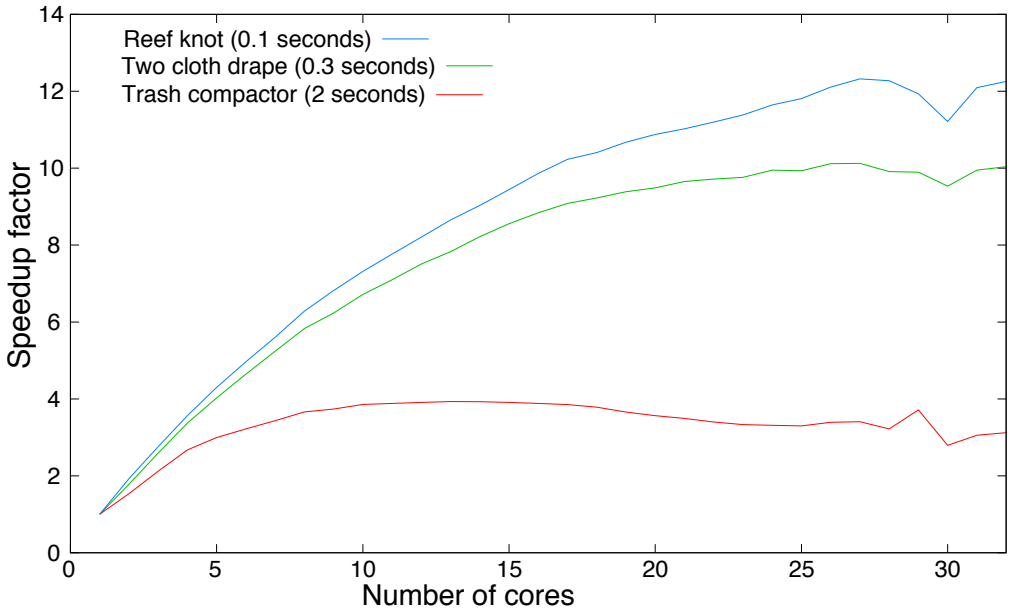


Figure 3.6: Scaling behavior of our method for three examples for different numbers of available cores. Speedup factor is relative to our method run on only a single core
. The reef knot is run for a short amount of time to show the behavior when the computation is dominated by material force updates. The trash compactor's poor scaling is due to the small problem size (less than a thousand degrees of freedom). The cloth drape is a more typical example of a large problem with many penalty contacts.

stored, and looking up that impulse during the velocity update randomly accesses memory. For processing internal forces, where the set of forces that affect each vertex is constant throughout the simulation, cache performance can be improved by partitioning the mesh into disjoint regions and processing each region in parallel (with special handling of the region boundaries), without intermediate storage. However, it is unclear how this strategy would extend to penalty force processing, and our initial experiments with partitioning the material forces resulted in only a modest (7%) performance improvement.

3.6.1 Choice of Rollback Window Size

Our framework introduces one additional parameter not present in the original algorithm: the duration of the rollback window R. Whereas choice of this parameter can have dramatic

effects on performance, it bears emphasis that the three guarantees are not compromised by any choice of R.

What is the optimal choice of R? Is it greater than zero, i.e. is there a point to rollback at all? In what follows, we give a theoretical argument that there exists a "sweet spot" for R, and study how the wall clock time of the benchmark examples vary with R.

Effect of R on number of rollbacks In a simulation with frequent collisions, increasing R increases the number of times a window rolls back on average, since during a large window it is more likely that the penalty forces that were added to fix one set of collisions will introduce a new set of collisions.

Effect of R on collision detection Consider an ideal simulation with frequent self-collisions uniformly distributed in time, and assume that the number of times the simulation rolls back does not change as R varies. Changing R changes the amount of time needed by collision detection for each rollback window. Fitting of the leaf nodes of the broad phase scales roughly as R, since computing the swept volumes of the simulation's triangles requires tracing the triangle's trajectory through the rollback window. The cost of the narrow phase is approximately proportional to $R \log R$, since the number of collisions is proportional to R and the cost of the divide-and-conquer separating slabs narrow phase is roughly logarithmic. Refitting interior nodes, and traversing the hierarchy, becomes more expensive as R increases (since traversal will reach the leaf level more often when more collisions are present) but has a non-trivial base constant cost independent of the size of the rollback window.

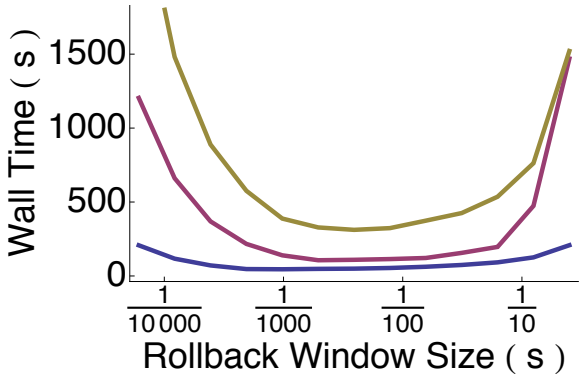


Figure 3.7: The total wall clock time, as a function of R, for the trash compactor (*blue*), two-cloth drape (*maroon*), and reef knot (*tan*). The "sweet spot" for R does not vary much between simulations and the run time of the simulations is insensitive to small perturbations of R.

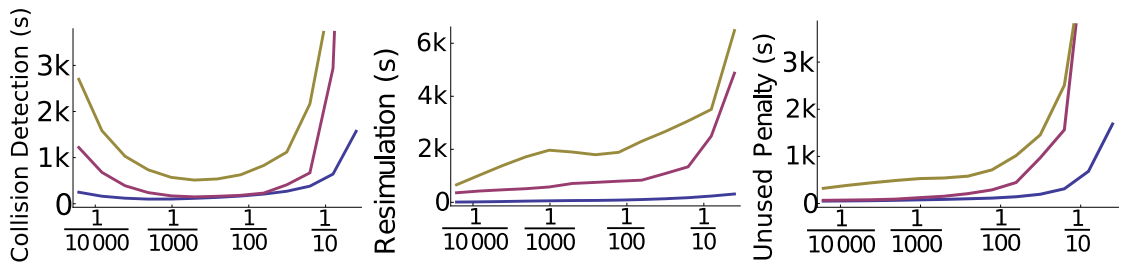


Figure 3.8: We measured, for the trash compactor (*blue*), two-cloth drape (*maroon*), and reef knot (*tan*), the CPU time wasted on collision detection (*left*), resimulation (*middle*), and unneeded penalty forces (*right*) as a function of R. As we expect, resimulation and unneeded penalty forces increase as R increases, while collision detection time has a sweet spot and grows large as R becomes too small.

Since the number of rollback windows in a simulation is inversely proportional to R, the total cost of the narrow phase is roughly $\log R$, whereas the overhead cost of BVH traversal decreases as R increases and grows unbounded as R shrinks. We therefore expect to minimize total cost at a sweet spot balancing these factors.

Effect of R on event processing The cost of event processing is proportional to the total window simulation time and resimulation time (assuming the cost of unnecessary penalty forces, etc. is negligible.) This simulation time is determined by the average number of times each window rolls back, which in turn increases with R. We thus expect wasted reprocessing work to increase with R.

Effect of R on unneeded penalty forces Unneeded penalty forces linger on the queue for two reasons: first, if multiple collisions are detected during a rollback window, it is possible that the penalty forces added to resolve one collision will, as a side-effect, also prevent the other collision, so that the forces added to prevent the second collision never act. Second, a penalty force might prevent a transient impact during one small part of a rollback window, and do nothing the remainder of the window. Both of these situations become more likely as R increases, so we expect the cost due to unneeded penalty forces to increase with increasing R.

Benchmark data We studied the effect of R on CPU time spent on collision detection, resimulation, and processing unneeded penalty layers, for three of our benchmark examples. Figure 3.8 plots this data. As expected, the cost of resimulating and of unneeded penalty forces increases with R, and the cost of collision detection forms a "U" shape, with the cost increasing as R becomes too small. In Figure 3.7 we plot the total wall clock time, as a function of R, for the same examples. The total wall clock time behaves similarly to collision detection time, confirming that rollback is indeed useful: performing collision detection over a window is significantly cheaper than doing so after every event is processed, and the increase in other waste is modest.

We also observe from Figure 3.7 that small perturbations of R near the sweet spot does not significantly change the wall clock time of the simulation; hence performance does not depend critically on pinpointing the exact sweet spot. For all timings in this paper we simply used $R = 0.003$, which seems to work well for all of our examples; with more study and analysis we hope to provide, in the future, a heuristic for automatically selecting R, and an adaptive algorithm for adjusting R over the course of a simulation.

3.6.2 Parallel AVIs

Recent exciting work Huang *et al.* [2007]; Pingali *et al.* [2011] has examined parallelization of AVIs for physical systems without contact, and we studied the possibility of incorporating this work into our framework. Unfortunately, the available parallelism of our event queue does not appear sufficiently high for graph-based out-of-order execution to be profitable. If we leave internal and penalty forces bucketed, we have that the internal force bucket is connected to every other event (since the internal forces affect every vertex), and each penalty layer event is connected to every other penalty layer so that the event dependency graph is complete and the worklist never contains more than one event. Nevertheless, we hope to explore the possibility of combining ACM and Galois in future work, particularly for simulations involving cloth-shell coupling or other scenarios where the stable timesteps of the internal forces vary widely and naively bucketing them is expensive.

3.7 Conclusion

The ACM framework offered unparalleled correctness and robustness guarantees, but at a steep performance cost relative to other popular methods for simulating cloth, shells, and deformable bodies. By replacing ACM's KDS-based collision detection paradigm with one based on Time Warp, we both dramatically improve its efficiency and allow it to be easily parallelized, for a total of two orders of magnitude speedup over Harmon et al. This speedup is a significant step towards ACM being viable for production.

The above speculative approach leaves open several avenues of immediate future work, such as incorporating parallel implementations of the broad phase collision detection algorithms (BVHs), adaptively selecting the rollback window size, etc, but also exposes aspects of the ACM framework that will become fundamental challenges to continuing efforts at optimizing its performance:

- First, with or without contact, time integration is a parabolic problem that is inherently cache-limited. The most basic possible stretching and bending stencils require examining one or two triangles' worth of position data, respectively, and in practice more sophisticated constitutive laws require fetching many times that much data in the form of stiffnesses, rest configurations, velocities, etc. The amount of computation performed on the data is relatively small compared to the amount of data read, limiting the benefits of parallelization. One possible solution would be to *implicitly* integrate material forces, trading frequent, cheap integration events for infrequent, expensive ones. Building on recent work Harmon *et al.* [2011] in this direction will be essential to developing more efficient, parallel implementations of ACM.

- Many of the advantages of the speculative approach to ACM depend on material force computation time dominating penalty force integration time, which is true of physical systems with sporadic or resting contact. Efficiently simulating systems with heavy, persistent contact, such as the late stages of the spinning rod experiments (figure 3.5), will require additional improvements to the algorithm. Again, the key factor is cache performance: reordering the mesh (section 3.5.3) improves spatial locality of the *material domain*, which speeds up integrating material forces, but does

not take advantage of spatial locality in *ambient space*, which is needed for efficient integration of penalty force events. Alternative data structures like a spatial hash could be used to speed up penalty force processing, at the cost of making material force processing less efficient. Developing hybrid data structures that are efficient in both regimes promises to further improve the performance of the algorithm.

Part II

Geometry of Thin Shells and Sheets

Chapter 4

Self-Supporting Masonry Structures

Vaulted masonry structures are among the simplest and at the same time most elegant solutions for creating curved shapes in building construction. For this reason they have been an object of interest since antiquity, and continue to be an active topic of research today.

Masonry is particularly interesting to the study of the geometry of surfaces since its extreme material properties allow us to make several simplifying assumptions. In particular, following Heyman 1966, the remainder of this chapter will assume:

- Masonry has no tensile strength, but the individual building blocks do not slip against each other (because of friction or mortar). On the other hand, their compressive strength is sufficiently high so that failure of the structure is by a sudden change in geometry and not by material failure.

- If a system of forces can be found which is in equilibrium with the load on the structure and which is contained within the masonry envelope then the structure will carry the loads, although the actual forces present may not be those postulated by that system (the *Safe Theorem*).

As will be shown in this chapter, these two assumptions transform a physics problem – the statics of volumetric structures – into a surface geometry problem. Insights about the

geometry of surfaces will translate into results on the shape of *self-supporting* masonry structures (structures that stand up under they own weight). They will permit us to build an efficient, interactive tool for exploring the space of self-supporting surfaces, lead to a particularly elegant algorithm for remeshing a given self-supporting surface into planar quadrilateral tile, and will allow us to characterize certain special families of surfaces that are always self-supporting; along the way, connections between different areas of physics, mathematics, and discrete differential geometry will be uncovered.

More specifically, the remainder of the chapter will

- Review the conditions for static equilibrium of structures satisfying the above material assumptions (section 4.2);
- Review the discretization of these equilibrium equations in terms of thrust networks (section 4.3);
- Describe four different geometric interpretations of the static equilibrium of self-supporting surfaces, and the connections between them (section 4.4);
- Present novel applications of these different views to form-finding and remeshing, and exhibits some results (section 4.5);
- Discuss possible extensions and future work (section 4.6).

More detail on each of these topics can be found in recent work by Vouga et al 2012.

4.1 Contributions

Understanding the static equilibrium of masonry structures is an active area of research; most relevant to the ideas presented in this chapter is the work on thrust network analysis Block and Ochsendorf [2007], summarized in section 4.3, and Maxwell's investigations of the connections between reciprogram diagrams and polyhedral surfaces Maxwell [1864]; Ash *et al.* [1988]. A more detailed survey of the earlier research in these areas, as well as other approaches to form-finding masonry structures, can be found in section 1.2.6.

Our contributions to the study of stable masonry structures include a new optimization algorithm for solving the inverse design problem (section 4.5): given an input unstable surface, our algorithm finds a nearby surface that is stable, and at the time of publication this algorithm was the fastest method for solving this difficulty nonlinear optimization problem, allowing interactive design. Secondly, we present a new understanding of equilibrium in terms of surface curvatures in isotropic space and in terms of "perfect" Wardetzky *et al.* [2007] discrete Laplace-Beltrami operators (section 4.4). This understanding allows us to characterize some special families of stable designs, and to find the unique planar quadrilateral remeshing of a stable surface that is still stable, a necessary step in constructing stable freeform surfaces built out of steel frames supporting glass panels.

Since the work presented here was published Vouga *et al.* [2012], it has served as the foundation for several followup investigations. de Goes et al 2013 formalize the geometry of stability in the language of Discrete Exterior Calculus, and explore the duality between self-supporting surfaces specified by edge weights, and Airy stress polytopes (see section 4.4.2) specified by vertex heights; for triangular meshes, this change of variables leads to improvements in the efficiency of design algorithms. Liu et al 2013 propose improving the quality and force flow through a self-supporting surface by smoothing the dual-isotropic relative curvature of the surface (see section 4.4.3). Panozzo et al 2013 describe a complete pipeline for constructing self-supporting structures out of blocks, which requires extra steps beyond the design and optimization described here.

4.2 Smooth Equilibrium

By the Safe Theorem, studying the stability of a volumetric masonry structure can be reduced to studying the static equilibrium of surfaces embedded within the masonry envelope. We will assume further that the surface is manifold, and that it can be modeled by a height field $s(x, y)$ defined in some planar domain Ω, i.e. that the surface does not "double over" on itself. We assume a vertical load density $F(x, y)$ over the top view Ω – usually F represents the structure's own weight. By definition this surface is self-supporting if and only if there exists a negative semi-definite (compressive) stress tensor σ over the surface whose

stresses are in equilibrium with the acting forces. Rewriting the equilibrium equations in plane coordinates (x, y), we have that such a stress tensor exists if and only if there exists a field $M(x, y) = -\sigma g^{-1}\sqrt{\det g}$ of 2×2 symmetric positive semidefinite matrices satisfying

$$\operatorname{div}(M\nabla s) = F, \quad \operatorname{div} M = 0, \tag{4.1}$$

where g is the induced metric

$$\begin{bmatrix} 1 + s_x^2 & s_x s_y \\ s_x s_y & 1 + s_y^2 \end{bmatrix}$$

and the divergence operator $\operatorname{div} \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = u_x + v_y$ acts on the columns of a matrix (see e.g. Fraternali [2010], Giaquinta and Giusti [1985]).

The condition $\operatorname{div} M = 0$, along with symmetry of M, can be viewed as integrability conditions on the components of M: in particular, writing

$$\widehat{H} = J^T H J$$

for J the ninety-degree rotation matrix $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, we have that $\widehat{H}$ is the Hessian of a real-valued function ϕ (the *Airy stress potential*) Green and Zerna [2002], and

$$M = \widehat{\nabla^2 \phi}.$$

If the domain Ω is simply connected, this relation holds globally. Negative semi-definiteness of M (or equivalently of $\widehat{M}$) is equivalent to *convexity* of the negative Airy potential $-\phi$, when interpreted as a height field over Ω in its own right.

In masonry structures two types of boundary conditions are typical: anchored boundaries (where the building is e.g. sunk into the ground with sufficient support to withstand arbitrary traction), where the boundary stresses are unconstrained, and *free boundaries*, where $\langle M\nabla s, \mathbf{n}\rangle = 0$, for $\mathbf{n}$ the boundary normal.

4.3 Discretization of Equilibrium

Following Block 2007, a self-supporting surface can be discretized by a mesh $\mathcal{S} = (V, E, F)$. We again assume vertical loads, and discretize them as force densities F_i associated with

vertices $\mathbf{v}_i$. The load acting on this vertex is then given by $F_i A_i$, where A_i is an area of influence (using a prime to indicate projection onto the xy plane, A_i is the area of the Voronoi cell of $\mathbf{v}_i'$ w.r.t. V'). We assume that stresses are carried by the edges of the mesh: the force exerted on the vertex $\mathbf{v}_i$ by the edge connecting $\mathbf{v}_i, \mathbf{v}_j$ is given by

$$w_{ij}(\mathbf{v}_j - \mathbf{v}_i), \quad \text{where} \quad w_{ij} = w_{ji} \geq 0.$$

The weights w_{ij} in these equations can be interpreted as axial force densities along the edges. The nonnegativity of the individual weights w_{ij} expresses the compressive nature of forces. Writing $\mathbf{v}_i = (x_i, y_i, s_i)$, the balance conditions at the vertices can be separated into vertical and horizontal components:

$$\sum\nolimits_{j\sim i} w_{ij}(x_j - x_i) = \sum\nolimits_{j\sim i} w_{ij}(y_j - y_i) = 0, \tag{4.2}$$

$$\sum\nolimits_{j\sim i} w_{ij}(s_j - s_i) = A_i F_i. \tag{4.3}$$

A mesh equipped with edge weights in this way is a discrete *thrust network* Block [2009]. Invoking the safe theorem, a masonry structure is self-supporting if we can find a thrust network with compressive forces entirely contained within the structure. In other words, for a given surface the $\mathbf{v}_i$ are known and the w_{ij} are unknown; the surface is self-supporting whenever a solution $\{w_{ij}\}$ to the above equations exist. Finding a self-supporting surface near one that is not amounts to solving for a simultaneous solution in $\mathbf{v}_i$ and w_{ij}.

4.4 Geometry of Equilibrium

Static equilibrium of both smooth and discrete self-supporting surfaces can be characterized geometrically in at least four related ways: in terms of reciprocal diagrams, convex Airy stress potentials, dual-isotropic relative mean curvature, or Laplace-like elliptic operators.

4.4.1 Reciprocal Diagrams

Let $\mathcal{S}'$ denote the *top view* of $\mathcal{S}$: its orthogonal projection onto the ground plane. Primed edge vectors in this section will similarly denote corresponding planar edges of the top view. Equations (4.2) have a geometric interpretation in terms of the top view $\mathcal{S}'$: since edge $\mathbf{e}_{ij}'$

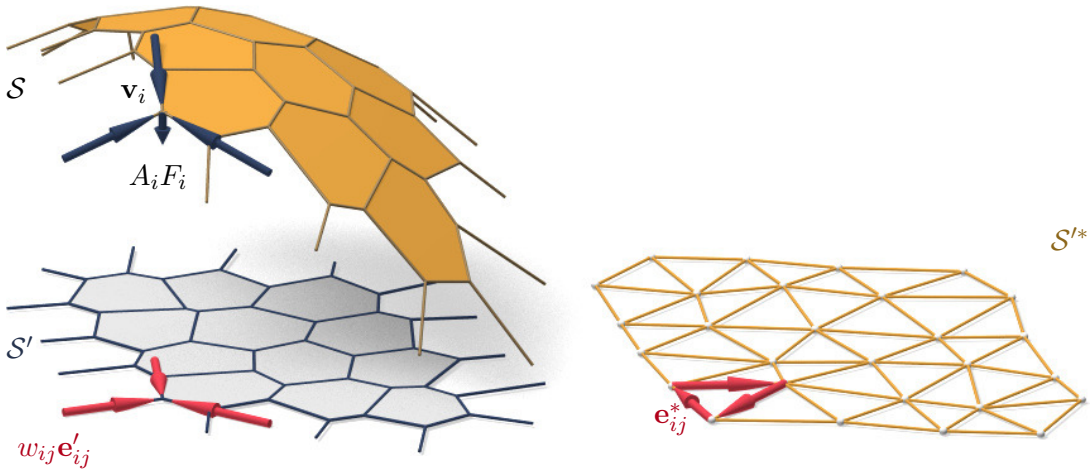


Figure 4.1: A thrust network $\mathcal{S}$ with dangling edges indicating external forces (left). This network together with compressive forces which balance vertical loads A_iF_i projects onto a planar mesh $\mathcal{S}'$ with equilibrium compressive forces $w_{ij}\mathbf{e}'_{ij}$ in its edges. Rotating forces by 90° leads to the reciprocal force diagram $\mathcal{S}'^*$ (right).

is given by

$$\mathbf{e}'_{ij} = \mathbf{v}'_j - \mathbf{v}'_i = (x_j, y_j) - (x_i, y_i),$$

it follows from equation (4.2) that vectors $w_{ij}\mathbf{e}'_{ij}$ summed around each vertex must add up to zero. Rotating them by 90 degrees, we see that likewise

$$\mathbf{e}'^*_{ij} = w_{ij} J \mathbf{e}'_{ij}, \quad \text{with} \quad J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},$$

form a closed cycle (see Figure 4.1). If the mesh $\mathcal{S}$ is simply connected, there exists an entire *reciprocal diagram* $\mathcal{S}'^*$ which is a combinatorial dual of $\mathcal{S}$, and which has edge vectors $\mathbf{e}'^*_{ij}$ Block and Ochsendorf [2007]. Its vertices (dual to the faces of $\mathcal{S}'$) will be denoted by $\mathbf{v}'^*_i$. As one special case, if $\mathcal{S}'$ is a Delaunay triangulation, then the corresponding Voronoi diagram is an example of a reciprocal diagram.

The reciprocal diagram can be viewed as the projection onto the plane of the *polar dual* of $\mathcal{S}$ with respect to the *Maxwell paraboloid* $z = \frac{1}{2}(x^2 + y^2)$, which maps the plane $z = \alpha x + \beta y + \gamma$ to the point $(\alpha, \beta, -\gamma)$ (see Fig. 4.2). Note that the polarity map can be applied to the smooth surface s as well, by mapping each point of s to the image of

its tangent plane under the polar map; we will revisit this idea below, in the section on dual-isotropic relative curvatures.

4.4.2 Airy Stress Polyhedron

In analogy to the smooth stress potential ϕ, we can construct a convex polyhedral "Airy stress potential" surface Φ with vertices $\mathbf{w}_i = (x_i, y_i, \phi_i)$ combinatorially equivalent to $\mathcal{S}$ by requiring that a primal face of Φ lies in the plane $z = \alpha x + \beta y + \gamma$ if and only if (α, β) is the corresponding dual vertex of $\mathcal{S}'^*$ (see Figure 4.2). Obviously this condition determines Φ up to vertical translation, and up to addition of linear functions, which can be viewed as translations of the reciprocal diagram $\mathcal{S}'^*$ in the plane.

Locally around each vertex, there exist choices of γ for which the neighboring Airy polydron faces "close up" Ash *et al.* [1988]; this integrability condition is exactly the equilibrium equation 4.2. If the top view is simply connected, the Airy polyhedron exists globally. The inverse procedure constructs a reciprocal diagram from Φ. This procedure works regardless of whether the forces are compressive, but like in the smooth case, an Airy mesh Φ is convex when the forces are compressive.

The vertices of Φ can be interpolated by a piecewise-linear function $\phi(x, y)$. It is easy to see that the derivative of $\phi(x, y)$ jumps by the amount $\|\mathbf{e}'^*_{ij}\| = w_{ij}\|\mathbf{e}'_{ij}\|$ when crossing over the edge $\mathbf{e}'_{ij}$ at right angle, with unit speed. This identifies Φ as the Airy polyhedron introduced by Fraternali *et al.* [2002] as a finite element discretization of the continuous Airy function (see also Fraternali [2010]).

4.4.3 Dual-isotropic Relative Curvature

Generally speaking, in the differential geometry of surfaces one considers the *Gauss map* σ from a surface S to a convex unit sphere ϕ by requiring that corresponding points have parallel tangent planes. Subsequently mean curvature H^{rel} and Gaussian curvature K^{rel} *relative to* ϕ are computed from the derivative $d\sigma$. Classically ϕ is the ordinary unit sphere $x^2 + y^2 + z^2 = 1$, so that σ maps each point to its unit normal vector.

We can instead write the Gauss map in terms of the polar duals S^*, ϕ^*, where it takes a particularly simple form, since points on the surfaces with parallel tangent planes map

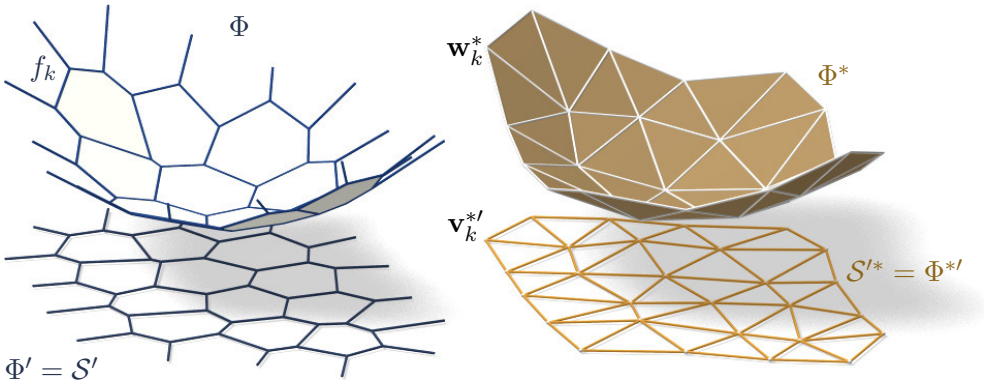


Figure 4.2: Airy stress potential Φ and its polar dual Φ^*. Φ projects onto the same planar mesh as $\mathcal{S}$ does, while Φ^* projects onto the reciprocal force diagram. A primal face f_k lies in the plane $z = \alpha x + \beta y + \gamma \iff$ the corresponding dual vertex is $\mathbf{w}_k^* = (\alpha, \beta, -\gamma)$.

under polarity to vertically-aligned points. If we know which point a plane is attached to, then the Gauss map is determined by the plane's gradient. So we simply write

$$\nabla\phi \overset{\sigma}{\longmapsto} \nabla s.$$

By moving along a curve $\mathbf{u}(t) = (x(t), y(t))$ in the parameter domain we get the first variation of tangent planes: $\frac{d}{dt}\nabla\phi|_{\mathbf{u}(t)} = (\nabla^2\phi)\dot{\mathbf{u}}$. This yields the derivative $(\nabla^2\phi)\dot{\mathbf{u}} \overset{d\sigma}{\longmapsto} (\nabla^2 s)\dot{\mathbf{u}}$, for all $\dot{\mathbf{u}}$, and the matrix of $d\sigma$ is found as $(\nabla^2\phi)^{-1}(\nabla^2 s)$. By definition, curvatures of the surface s *relative* to ϕ are found as

$$K_s^{\text{rel}} = \det(d\sigma) = \tfrac{\det\nabla^2 s}{\det\nabla^2\phi},$$
$$H_s^{\text{rel}} = \tfrac{1}{2}\operatorname{tr}(d\sigma) = \tfrac{1}{2}\operatorname{tr}\left(\tfrac{M}{\det\nabla^2\phi}\nabla^2 s\right) = \tfrac{\Delta_\phi s}{2\det\nabla^2\phi}.$$

The Maxwell paraboloid $\phi_0(x, y) = \frac{1}{2}(x^2 + y^2)$ is the canonical unit sphere of isotropic geometry; unsurprisingly, given its horizontal symmetry and privileged vertical direction, using it instead of the Euclidean sphere will prove particularly useful for analyzing the static equilibrium of surfaces under vertical load.

Curvatures relative to ϕ_0 will be denoted by the symbols H, K instead of $H^{\text{rel}}, K^{\text{rel}}$.

The observation

$$\Delta_\phi \phi = \operatorname{tr}(M \nabla^2 \phi) = \operatorname{tr}(\widehat{\nabla^2 \phi} \nabla^2 \phi) = 2 \det \nabla^2 \phi$$

together with the formulas above implies

$$K_s = \det \nabla^2 s, \;\; K_\phi = \det \nabla^2 \phi \implies H_s^{\text{rel}} = \frac{\Delta_\phi s}{2K_\phi} = \frac{\Delta_\phi s}{\Delta_\phi \phi}.$$

Summarizing the formulas above, the equilibrium condition (4.1) can be written *purely in terms of the geoemtry of s and ϕ* as

$$2K_\phi H_s^{\text{rel}} = \Delta_\phi s = F. \tag{4.4}$$

This curvature relation has a discrete analogue. A general theory of curvatures of polyhedral surfaces with respect to a polyhedral unit sphere was proposed by Pottmann *et al.* [2007]; Bobenko *et al.* [2010], and its dual complement in isotropic geometry was elaborated on in Pottmann and Liu [2007]. Mean curvature of a self-supporting surface $\mathcal{S}^*$ relative to its discrete Airy stress polytope Φ^* can be computed from areas and mixed areas of their polar duals:

$$H^{\text{rel}}(\mathbf{v}_i) = \frac{A_i(\mathcal{S}, \Phi)}{A_i(\Phi, \Phi)}, \quad \text{where}$$
$$A_i(\mathcal{S}, \Phi) = \frac{1}{4} \sum_{k : f_k \in \text{1-ring}(\mathbf{v}_i)} \det(\mathbf{v}_k'^*, \mathbf{w}_{k+1}'^*) + \det(\mathbf{w}_k'^*, \mathbf{v}_{k+1}'^*).$$

The prime denotes the projection into the xy plane, and summation is over those dual vertices which are adjacent to $\mathbf{v}_i$. Replacing $\mathbf{v}_k^*$ by $\mathbf{w}_k^*$ yields $A_i(\Phi, \Phi) = \frac{1}{2} \sum \det(\mathbf{w}_k'^*, \mathbf{w}_{k+1}'^*)$, and it can be shown Vouga *et al.* [2012] that this expression can be rewritten as

$$H^{\text{rel}}(\mathbf{v}_i) = \frac{\sum_{j \sim i} w_{ij}(s_j - s_i)}{\sum_{j \sim i} w_{ij}(\phi_j - \phi_i)} = \frac{\Delta_\phi s}{\Delta_\phi \phi}\Big|_{\mathbf{v}_i}. \tag{4.5}$$

In order to discretize (4.4), we also need a discrete Gaussian curvature, usually defined as a quotient of areas which correspond under the Gauss mapping. We define

$$K_\Phi(\mathbf{v}_i) = \frac{A_i(\Phi, \Phi)}{A_i},$$

where A_i is the Voronoi area of vertex $\mathbf{v}_i'$ in the projected mesh $\mathcal{S}'$ used in (4.3).

In direct analogy to the geometric characterization of smooth equilibrium (4.4), a simply-connected mesh $\mathcal{S}$ with vertices $\mathbf{v}_i = (x_i, y_i, s_i)$ can be put into static equilibrium with vertical nodal forces $A_i F_i$ if and only if there exists a combinatorially equivalent mesh Φ with planar faces and vertices (x_i, y_i, ϕ_i), such that curvatures of $\mathcal{S}$ relative to Φ obey

$$2K_\Phi(\mathbf{v}_i)H^{\mathrm{rel}}(\mathbf{v}_i) = F_i \tag{4.6}$$

at every interior vertex and every free boundary vertex $\mathbf{v}_i$. $\mathcal{S}$ can be put into compressive static equilibrium if and only if there exists a convex such Φ.

4.4.4 Stress Laplacian

In the smooth setting, the expression $\operatorname{div}(M\nabla s)$ can be interpreted as a linear operator $\Delta_\phi s$, with ϕ, as above, the Airy stress potential generating M. Note that this operator is self-adjoint, elliptic, and that $\operatorname{div} m = 0$ is precisely the condition needed for Δ_ϕ to have linear functions in its kernel. The balance condition (4.1) may be written as $\Delta_\phi s = F$.

This Laplace-like elliptic operator has a discrete counterpart in terms of the weights w_{ij}: we can define a graph Laplacian (acting on vertex-based functions) by

$$\Delta_\Phi s(\mathbf{v}_i) = \sum\nolimits_{j\sim i} w_{ij}(s_j - s_i).$$

This operator is a perfect discrete Laplacian in the sense of Wardetzky *et al.* [2007], since it is symmetric by construction, Equation (4.2) implies linear precision for the planar "top view mesh" $\mathcal{S}'$ (i.e., $\Delta_\Phi f = 0$ if f is a linear function), and $w_{ij} \geq 0$ ensures semidefiniteness and a maximum principle for Δ_Φ-harmonic functions. Equation (4.3) can be written as $\Delta_\Phi s = AF$.

When considering discrete thrust networks as the discretizations of continuous self-supporting surfaces, the following question is important: For a given smooth surface $s(x, y)$ with stress potential ϕ, does there exist a polyhedral surface $\mathcal{S}$ in equilibrium approximating $s(x, y)$, whose top view is a given planar mesh $\mathcal{S}'$? We restrict our attention to triangle meshes, where planarity of the faces of the discrete stress surface Φ is not an issue. Equivalently, we can ask:

- Does $\mathcal{S}'$ have a reciprocal diagram whose corresponding Airy polyhedron Φ approximates the continuous Airy potential ϕ? (on a local patch of the surface, if it is not simply connected.)
- Does $\mathcal{S}'$ possess a "perfect" discrete Laplace-Beltrami operator Δ_Φ in the sense of Wardetzky et al. 2007 whose weights are the edge length scalars of such a reciprocal diagram?

From Wardetzky *et al.* [2007] we know that perfect Laplacians exist only on regular triangulations which are projections of convex polyhedra. On the other hand, previous sections show how to appropriately retriangulate: Let Φ be a triangle mesh convex hull of the vertices $(x_i, y_i, \phi(x_i, y_i))$, where (x_i, y_i) are vertices of $\mathcal{S}'$. Then its polar dual Φ^* projects onto a reciprocal diagram with positive edge weights, so Δ_ϕ has positive weights, and the vertices (x_i, y_i, s_i) of $\mathcal{S}$ can be found by solving the discrete Poisson problem $(\Delta_\Phi s)_i = A_i F_i$.

It is plausible that the discrete stress Laplacian Δ_Φ converges to Δ_ϕ for reasonable refinements of $\mathcal{S}$; This would imply that solving the discrete Poisson equation leads to a mesh approximating its continuous counterpart $s(x, y)$, and we have convergence as the sampling density increases. A rigorous analysis is a topic for future research.

4.5 Application and Results

4.5.1 Interactive Form-Finding

Consider the problem of taking a given reference mesh, say $\mathcal{R}$, and finding a combinatorially equivalent mesh $\mathcal{S}$ in static equilibrium approximating $\mathcal{R}$. The loads on $\mathcal{S}$ include user-prescribed loads as well as the dead load caused by the mesh's own weight. Conceptually, finding $\mathcal{S}$ amounts to minimizing some formulation of distance between $\mathcal{R}$ and $\mathcal{S}$, subject to constraints (4.2), (4.3), and $w_{ij} \geq 0$. For any choice of distance this minimization will be a nonlinear, non-convex, inequality-constrained variational problem. Black-box solvers Wächter and Biegler [2006] perform well for surfaces without complex geometry or for polishing reference meshes close to self-supporting, but fail to converge in reasonable time for more complicated shapes. We therefore proposed the following specialized, staggered

linearization for solving the optimization problem:

0. Start with an initial guess $\mathcal{S} = \mathcal{R}$.
1. Estimate the self-load on the vertices of $\mathcal{S}$, using their current positions.
2. Fixing $\mathcal{S}$, locally fit an associated stress surface Φ.
3. Alter positions $\mathbf{v}_i$ to improve the fit.
4. Repeat from Step 1 until convergence.

Remark: This staggered approach shares several advantages of solving the full nonlinear problem: a nearby self-supporting surface is found given only a suggested reference shape, without needing to single one of the many possible top view reciprocal diagrams or needing to specify boundary tractions – these are found automatically during optimization. Although providing an initial top view graph with good combinatorics remains important, by not fixing the top view our approach allows the thrust network to slide both vertically and tangentially to the ground, essential to finding faithful thrust networks for surfaces with free boundary conditions.

Step 1: Estimating Self-Load. The dead load due to the surface's own weight depends not only on the top view of $\mathcal{S}$, but also on the surface area of its faces. To avoid adding nonlinearity to the algorithm, we estimate the load coefficients F_i at the beginning of each iteration, and assume they remain constant until the next iteration. We estimate the load $A_i F_i$ associated with each vertex by calculating its Voronoi surface area on each of its incident faces (note that this surface area is distinct from A_i, the vertex's Voronoi area on the top view), and then multiplying by a user-specified surface density ρ.

Step 2: Fit a Stress Surface. In this step, we fix $\mathcal{S}$ and try to fit a stress surface Φ subordinate to the top view $\mathcal{S}'$ of the primal mesh. We do so by searching for dihedral angles between the faces of Φ which minimize, in the least-squares sense, the error in force equilibrium (4.6) and local integrability of Φ. Doing so is equivalent to minimizing the squared residuals of Equations (4.3) and (4.2), with the positions held fixed. We define the

equilibrium energy

$$E = \sum_i \left\| \begin{pmatrix} 0 \\ 0 \\ A_i F_i \end{pmatrix} - \sum_{j \sim i} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) \right\|^2, \tag{4.7}$$

where i runs through interior and free boundary vertices, and solve

$$\min_{w_{ij}} E, \quad \text{s.t. } 0 \leq w_{ij} \leq w_{\max}. \tag{4.8}$$

Here $w_{\max}$ is an optional maximum weight we are willing to assign (to limit the amount of stress in the surface). This convex, sparse, box-constrained least-squares problem Friedlander [2007] always has a solution. If the objective is 0 at this solution, $\mathcal{S}$ is self-supporting – we are done. Otherwise, $\mathcal{S}$ is not self-supporting and its vertices must be moved.

Step 3: Alter Positions. In the previous step we fit as best as possible a stress surface Φ to $\mathcal{S}$. There are two possible kinds of error with this fit: the faces around a vertex (equivalently, the reciprocal diagram) might not close up; and the resulting stress forces might not be exactly in equilibrium with the loads. These errors can be decreased by modifying the top view and heights of $\mathcal{S}$, respectively. It is possible to simply solve for new vertex positions that put $\mathcal{S}$ in static equilibrium, since Equations (4.2) and (4.3) with w_{ij} fixed form a square linear system that is typically nonsingular.

While this approach would yield a self-supporting $\mathcal{S}$, this mesh is often far from the reference mesh $\mathcal{R}$, since any local errors in the stress surface from Step 2 amplify into global errors in $\mathcal{S}$. We propose instead to look for new positions that decrease the imbalance in the stresses and loads, while also penalizing drift away from the reference mesh:

$$\min_{\mathbf{v}} E + \alpha \sum_i \langle \mathbf{n}_i, \mathbf{v}_i - \mathbf{v}_i^0 \rangle^2 + \beta \left\| \mathbf{v} - \mathbf{v}_P^0 \right\|^2,$$

where $\mathbf{v}_i^0$ is the position of the i-th vertex at the start of this step of the optimization, $\mathbf{n}_i$ is the starting vertex normal (computed as the average of the incident face normals), $\mathbf{v}_P^0$ is the projection of $\mathbf{v}^0$ onto the reference mesh, and $\alpha > \beta$ are penalty coefficients that are decreased proportionally to the decrease in E at every iteration of Steps 1–3. $\alpha = 1$ and $\beta = 0.1$ worked well as initial values in several examples. The second term allows $\mathcal{S}$ to slide over itself (if doing so improves equilibrium) but penalizes drift in the normal direction.

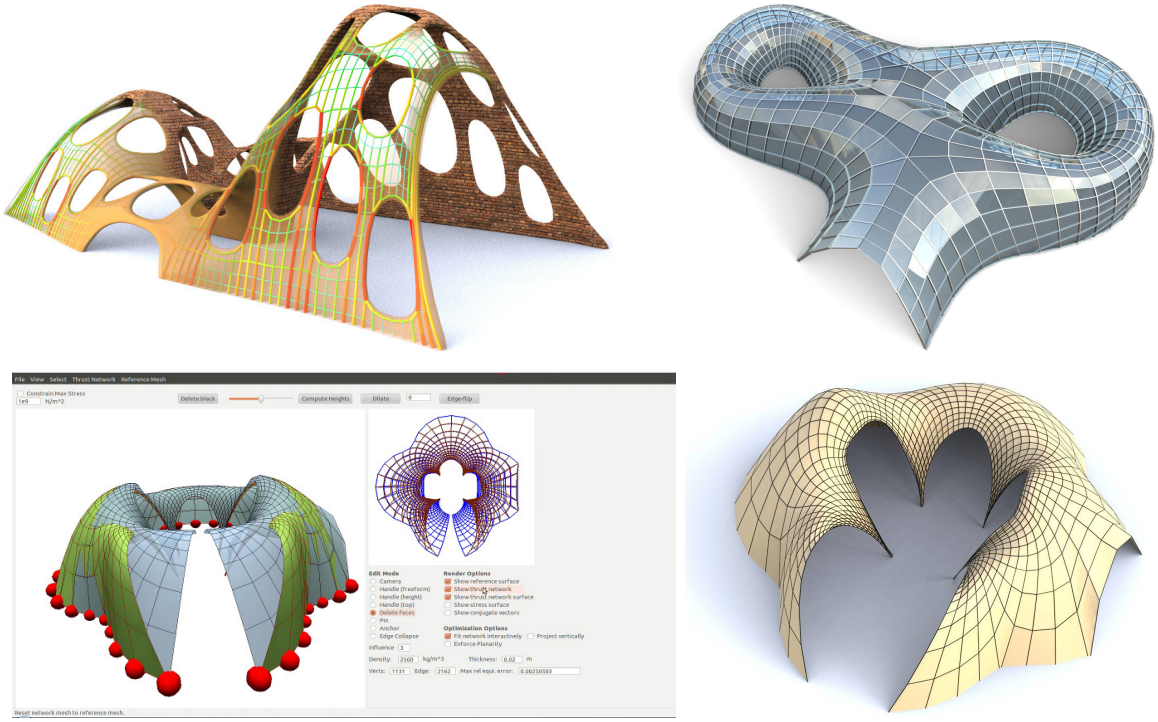


Figure 4.3: *Top left:* The stability of this surface, designed using our interactive tool, is not obvious at first glance since it is riddled with holes. Existence of an embedded surface in static equilibrium, shown as a wireframe, certify stability. Warmer colors represent higher stresses. *Top right:* Another example of a stable structure designed using our interactive design tool. *Bottom*: A screenshot of our tool *(left)* during design of a stable surface *(right)*. The user edits the blue mesh, while the tool interactively finds the nearby stable green surface.

The third term, weaker than the second, regularizes the optimization by preventing large drift away from the reference surface or excessive tangential sliding.

This algorithm is efficient enough to allow interactive editing of self-supporting surfaces; Vouga et al 2012 discuss the algorithm and editing tool in greater detail, and provide additional results and timing data. Figure 4.3 shows some representative examples of surfaces designed using this algorithm.

4.5.2 PQ Remeshing

Meshes with *planar* faces are of particular interest in architecture. The characterization of equilibrium in terms of relative curvatures to the Airy polyhedron leads to an algorithm for remeshing a given thrust network in equilibrium into a quad mesh with planar faces that is again in equilibrium. If this mesh is realized as a steel-glass construction, it is self-supporting in its beams alone, with no forces exerted on the glass. The beams constitute a self-supporting structure which is in perfect force equilibrium (without moments in the nodes) if only the deadload is applied. (In such constructions, the restriction that the internal forces are compressive does not apply.)

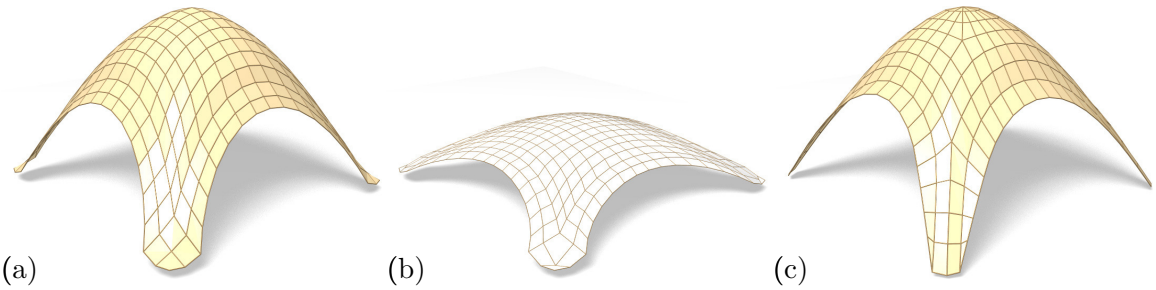


Figure 4.4: Directly enforcing planarity of the faces of even a very simple self-supporting quad-mesh vault (a) results in a surface far removed from the original design (b). Starting instead from a remeshing of the surface with edges following relative principal curvature directions yields a self-supporting, PQ mesh far more faithful to the original (c).

Taking an arbitrary non-planar quad mesh and attempting naive, simultaneous enforcement of planarity and static equilibrium – either by staggering a planarity optimization step every outer iteration, or adding a planarity penalty term to the position update – does not yield good results, as shown in Figure 4.4. Indeed, as we will see later in this section, such a planar perturbation of a thrust network is not expected to generally exist.

Consider a planar quad mesh $\mathcal{S}$ with vertices $\mathbf{v}_{ij} = (x_{ij}, y_{ij}, s_{ij})$ which approximates a given continuous surface $s(x, y)$. It is known that $\mathcal{S}$ must approximately follow a network of conjugate curves in the surface (see e.g. Liu *et al.* [2006a]); for curve networks on a smooth surface s, planarity requires that $(\mathbf{a}_1)^T \nabla^2 s \, \mathbf{a}_2 = 0$ where $\mathbf{a}_1$ and $\mathbf{a}_2$ are the tangent vectors of the curves at a point of intersection.

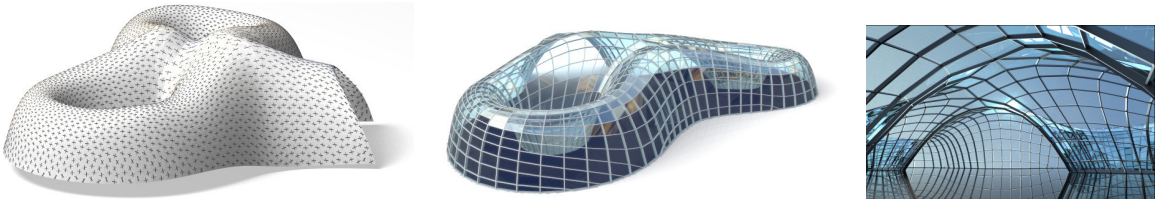


Figure 4.5: Planar quad remeshing of a self-supporting surface. Left: Illustrations of the relative principal direction vector fields. Center and Right: The result of optimization is a self-supporting PQ mesh, which guides a moment-free steel/glass construction.

In addition to the mesh $\mathcal{S}$ approximating the surface $s(x, y)$, the corresponding polyhedral Airy surface Φ must approximate $\phi(x, y)$; thus we get the conditions

$$(\mathbf{a}_1)^T \nabla^2 s \ \mathbf{a}_2 = (\mathbf{a}_1)^T \nabla^2 \phi \ \mathbf{a}_2 = 0.$$

$\mathbf{a}_1, \mathbf{a}_2$ are therefore eigenvectors of $(\nabla^2\phi)^{-1}\nabla^2 s$, and so are the principal directions of the surface $s(x, y)$ relative to $\phi(x, y)$.

In the discrete case, where s, ϕ are not given as continuous surfaces, but are represented by a mesh in equilibrium and its Airy mesh, we use the techniques of Schiftner 2007 and Cohen-Steiner and Morvan 2003 to approximate the Hessians $\nabla^2 s$, $\nabla^2\phi$, compute principal directions as eigenvectors of $(\nabla^2\phi)^{-1}\nabla^2 s$, and subsequently find meshes $\mathcal{S}, \Phi$ approximating s, ϕ which follow those directions. Global optimization can now polish $\mathcal{S}, \Phi$ to a valid thrust network with discrete stress potential, where before it failed: we do so by taking the planarity energy $\sum_f (2\pi - \theta_f)^2$, where the sum runs over faces and θ_f is the sum of the interior angles of face f, linearizing it at every iteration, and adding it to the objective function of the position update. Convexity of Φ ensures that $\mathcal{S}$ is self-supporting.

Note that for each Φ, the relative principal curvature directions give the *unique* curve network along which a planar quad discretization of a self-supporting surface is possible. Other networks won't work (see Figure 4.4). Figure 4.5.2 illustrate the result of applying this procedure to one example surface.

4.5.3 Special Families

Given a self-supporting thrust network $\mathcal{S}$ with stress surface Φ, which vertical perturbation $\mathcal{S} + \mathcal{R}$ is self-supporting, with the same loads as $\mathcal{S}$? We assume $\mathcal{S}' = \mathcal{R}' = \Phi'$, i.e. all meshes involved have the same top view, and arithmetic operations refer to the respective z coordinates s_i, r_i, ϕ_i of vertices.

The condition of equal loads is then $\Delta_\phi(s + r) = \Delta_\phi s$, in terms of Laplacians, or $H^{\text{rel}}_{\mathcal{S}} = H^{\text{rel}}_{\mathcal{S}+\mathcal{R}}$, in terms of mean curvature, and is equivalent to $\Delta_\phi r = 0$, i.e., $H^{\text{rel}}_{\mathcal{R}} = 0$. So $\mathcal{R}$ is a *minimal surface* relative to Φ. While in the triangle mesh case there are enough degrees of freedom for nontrivial solutions, the case of planar quad meshes is more intricate, and is discussed in detail in Vouga *et al.* [2012].

4.6 Discussion and Future Work

The above work connects the physics of surfaces and structures in static equilibrium under gravity to geometry. These geometric interpretations of equilibrium have straightforward discretizations, and led to new algorithms for efficient form-finding and PQ remeshing. The investigation also leaves open several avenues for future work:

- The space of self-supporting surfaces is not yet fully understood, in either the continuous or discrete settings. Does there exist a self-supporting structure given boundary conditions and loads? How many? Given a continuous self-supporting surface, what can be proven about its approximation by discrete ones? One promising avenue for resolving the latter question is combining the stress Laplacian view of equilibrium with recent "no free lunch" theorems on existence of discrete Laplace-Beltrami operators Wardetzky *et al.* [2007].

- Preliminary investigations suggest there are connections between the Airy stress polytope and the vertex weights of generalized discrete Hodge-star operators Mullen *et al.* [2011]; these connections could lead to a new computational method for designing self-supporting surfaces that is more efficient for thrust networks with fewer vertices than edges (such as regular triangle meshes).

- The existing model and discretization assume that the self-supporting surface is a height field over the plane, and that forces acting on the surface are vertical. Generalizing to height fields over arbitrary curved manifolds would lead to a computational method for designing self-supporting surfaces with more general loads, such as wind load.

Chapter 5

Developable Surfaces

The physical behavior of an elastic surface of small thickness h undergoing large deflections is notoriously complex and difficult to model computationally Chapelle and Bathe [1998], and is fundamentally different for surfaces of different intrinsic curvature character. An interesting special case are (piecewise) flat, *developable* surfaces: surfaces that can be unrolled onto the plane without stretching them. Such surfaces are locally characterized by having zero Gaussian curvature K at points where curvature is defined; equivalently, at those points at least one principal curvature direction is zero. Simple examples of developable surfaces include pieces of the plane, cylinders, and cones.

The membrane forces of an initially flat elastic plate, and more generally, initially developable surfaces, are given by the Föpple-von Kármán equations Landau *et al.* [1986], and can be decomposed into contributions from two terms:

- stretching forces, depending only on the intrinsic deformation of the plate, and proportional to the thickness h;
- bending forces, proportional to h^3, depending on the extrinsic curvature of the surface.

In the thin limit $h \to 0$, the stretching force dominates the bending force, and as such, thin surfaces bend much more readily than they stretch. This phenomenon is familiar to us in daily life, since when we crumple paper or crush an aluminum can, these materials crease and wrinkle but keep their intrinsic shape. In the limit of infinitesimal thickness, the surface remains perfectly developable, forming a piecewise linear Yoshimura pattern (see inset). For

surfaces of finite thickness, some stretching does occur. Often, this stretching localizes at ridges and cone points Boudaoud *et al.* [2000]; DiDonna [2002], with very little stretching away from these singular features, but in some cases diffuse stretching throughout the surface is observed Schroll *et al.* [2011]. Fully understanding which type of behavior occurs for which boundary conditions and loads, and how to efficiently represent and simulate surfaces exhibiting low, diffuse stress, remains an active area of research; in this chapter we will focus on the latter case, where stress is localized entirely along one- and two-dimensional singularities. For this regime, I propose modeling the deformation and crumpling of the thin elastic sheet using discrete, exactly-developable surfaces. Such a reduced representation of the surface would have several advantages: a discrete developable surface is designed to allow the sharp creases that arise during crumpling; a model that does not stretch can be accurately simulated using much larger, computationally-efficient elements, since the high-frequency modes of the stiff stretching force do not need to be resolved; and there is no danger of the *membrane locking* that plagues many lower-order finite shell elements, where the stretching forces artificially interfere with the surface's ability to bend.

The remainder of the chapter describes a collaboration Solomon *et al.* [2012] exploring one promising kinematic representation of discrete developable surfaces, and a formulation of bending energy for such surfaces. I also discuss how these energies might be augmented to take into account the stretching energy localized at the creases, and possible alternative, more computationally efficient formulations of the discrete developability conditions.

5.1 Contributions

To the previous work on representing and designing discrete approximations to developable surfaces (see section 1.2.9.2), we contribute several new components. We describe a novel bending energy (section 5.4) that discretizes the smooth mean curvature bending energy over discrete ruled patched, and discuss a workaround for dealing with cone singularities where neither discrete nor smooth curvature is well-defined (section 5.4.1). We propose a new, interactive design framework that, beyond enforcing kinematic developability constraints, uses this energy, and preliminary work on refining discrete developable surfaces

through ruling subdivision (section 5.5.2), to solve for the shape of developable thin surfaces containing both creases and smoothly curved patches and subject to user-specified boundary conditions.

5.2 Smooth Developable Surfaces

Developable surfaces have a variety of fascinating properties, see for instance Carmo [1976] or Pottmann and Wallner [2010] for a general discussion. Developable surfaces are characterized by having vanishing Gaussian curvature, that is, they are locally isometric to the Euclidean plane. Any smooth developable surface Σ locally is comprised of a one-parameter family of straight line segments called *rulings* that do not cross and run across the surface. Likewise, there exists a one-parameter family of smooth curves on Σ that run orthogonally to the rulings. Any curve γ in this family has the property that the orthonormal frame formed by (i) the unit tangent of γ, (ii) the (properly oriented) unit ruling direction, and (iii) the surface normal is a so-called natural (or Bishop) frame Bishop [1975]. This observation can be reversed: Consider a smooth space curve $\gamma(s)$ equipped with its orthonormal natural frame (T, U, V). Choosing $\epsilon > 0$ small enough yields a developable surface

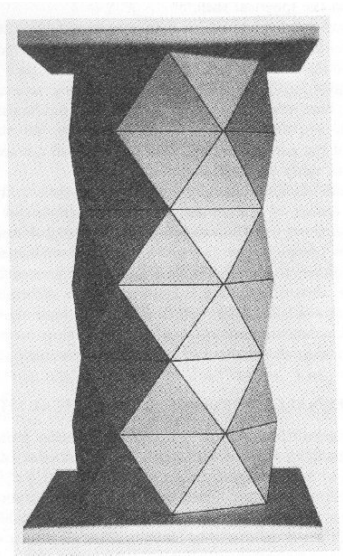
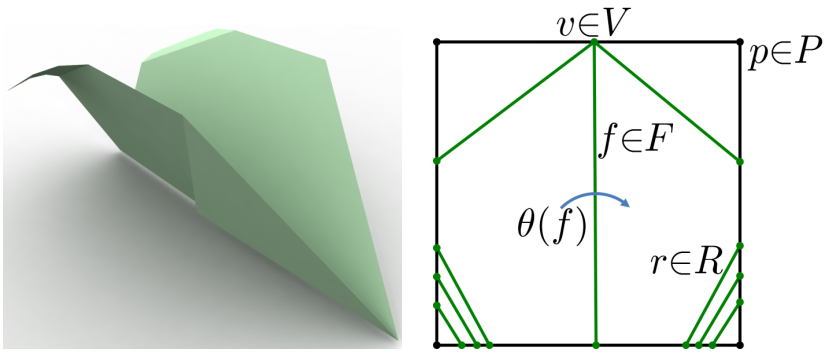


Figure 5.1: An infinitely thin surface deforms into a Yoshimura diamond pattern under axial compression. Photo by Khurram Wadee.

$$\Sigma(s,t) := \gamma(s) + tU(s)$$

for all $t \in (-\epsilon, \epsilon)$, with rulings formed by U and surface normals along γ formed by V.

In this case, the curvature of the resulting surface conveniently can be described in terms of the framed curve. Since it is perpendicular to the flat rulings, γ runs along a principal curvature line with normal curvature equal to $T' \cdot V$; this normal curvature is in turn equal to the mean curvature of Σ since the other principal curvature must be exactly zero. Likewise, for *fixed* t consider the curve $\gamma_t(s) = \Sigma(s,t)$. Re-parameterizing γ_t to arc-



(a) (b)

Figure 5.2: (a) A simple developable surface with two-dimensional configuration (b).

length, there is a natural frame for γ_t from which one can infer the mean curvature at $\Sigma(s,t)$ from the curvature of γ_t(s). Although we do not explicitly derive the requisite expressions here, this discussion serves as a motivation for the case of discrete developable surfaces, where (discrete) mean curvatures can be inferred from polygonal analogs of γ.

5.3 Discrete Developable Surfaces

As described above, we can think of a smooth developable surface as being composed of families of ruling lines; in addition to these rulings, a piecewise developable surface, such as results from crumpling a piece of paper, also contains sharp creases or *ridge lines* along which localized stretching occurs – on a piece of paper, we see this stretching as plastic damage to the paper's fibers (crinkles). Under the assumption of no diffuse stress, these ridge lines bound otherwise developable patches. We can think of rulings as lines along which the surface *infinitesimally* bends, and ridge lines as those along which the surface is bent some finite amount.

Although we are trying to approximate surfaces with smoothly curved, we think of discrete developable surfaces as pieces of origami that are folded from a sheet of paper without tearing or gluing. While such a discrete structure is piecewise flat, we do not view our surface as a collection of planar segments bordered by sharp folds. Instead, discrete folds are used to represent *both* rulings and crease lines. Those regions where external forces or other spatial constraints would naturally lead to curved configurations of the

smooth surface are approximated on a discrete developable surface with a *finite* number of rulings. Kinematically, we do not differentiate between discrete rulings and creases, but each type of fold has a different bending energies associated to it. Rulings (but not creases) can also be subdivided to yield, in the limit of refinement, a smooth piecewise developable surface, as will be described below.

Formally, in our setup, a discrete developable surface is comprised of the following parts:

- A *polygonal domain* $P \subset \mathbb{R}^2$ with corners $p_1, \ldots, p_m$. We assume that the edges of P do not self-intersect. For notational convenience, we let P include its own boundary.
- A collection of vertices $V = \{v_1, \ldots, v_n\} \subset P$, which can reside both on ∂P and in int P.
- The embedding of those vertices $Q = \{q_1, \ldots, q_n\}$ into $\mathbb{R}^3$.
- A collection of *folds* $F \subseteq V \times V$ representing line segments contained within P parameterized by their starting and ending vertices.
- A subset $R \subseteq F$ of *rulings* representing the interior of smoothly-bent pieces.
- A folding angle $\Theta : F \to \mathbb{R}$ associated with each fold; for a given fold $f = (v_i, v_j) \in F$ we denote its associated folding angle as θ_f.

Figure 5.2 labels all these components on a simple two-dimensional configuration.

Degrees of freedom (DOFs) Notice that there is considerable freedom in choosing which of the above quantities to specify as degrees of freedom, since many of the above parts depend on each other. One natural set of degrees of freedom are the two- and three-dimensional positions of the vertices V and Q; compatibility constraints then couple the distances between corresponding pairs of vertices, and the folding angles are dependent functions of the positions. Section 5.6 describes in more detail this representation, along with some of its pros and cons.

For a given domain P, there is another possibility: the configuration space of discrete developable surfaces can be parameterized, modulo rigid motion, by the graph topology, the positions of the vertices V, and the folding angles Θ of the fold lines. This is the

representation explored in my recent collaboration Solomon *et al.* [2012] and that will be described here. Note that this set of DOFs is distinguished from those used in the origami literature (see Tachi [2009b] and references therein) in that we consider variable, rather than fixed, vertex positions. Notice also that these degrees of freedom are entirely quantities associated to the planar domain P; the embedding of the surface is determined by these planar quantities.

To that essential set of minimal variables, it will be convenient to augment a redundant set of Euclidean transformations E mapping each face of P to its realization in $\mathbb{R}^3$. While these additional variables do not add expressive power to the discretization, they simplify the implementation of positional constraints and user manipulation in three-space.

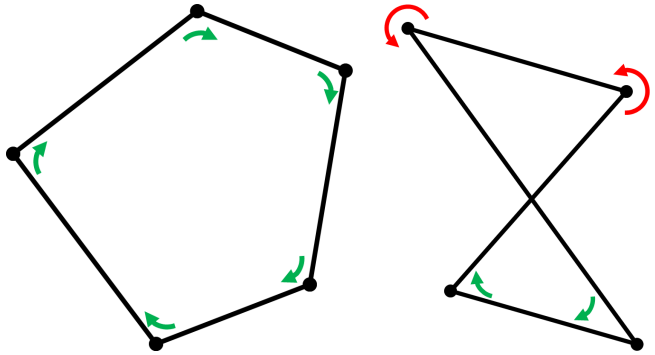
Boundary conditions and constraints Fold lines and prescribed bending angles Θ are given as user input or through the subdivision schemes described in Section 5.5.2. A challenge is to ensure developability, which corresponds to a number of *constraints* on V, F, and Θ that must be obeyed by admissible configurations.

5.3.1 Constraints

A number of constraints are needed to ensure that every configuration corresponds to a discrete developable surface with a realizable embedding in $\mathbb{R}^3$. These constraints can be divided into three groups:

- Geometric constraints on the vertices and edges of P: edges must intersect only at vertices, and vertices must remain within or on the boundary of the domain ∂P. (Section 5.3.2)

- Compatibility constraints imposed by 3D realizability: around every interior vertex, gluing together adjacent faces at their common edge with that edge's folding angle must yield a surface that locally "closes up." Moreover, the Euclidean transformations E must be compatible with the other degrees of freedom, so that neighboring faces meet seamlessly in 3D and at the correct folding angles. (Section 5.3.3)



(a) (b)

Figure 5.3: Polygons satisfying (a) and violating (b) the interior angle constraint.

- User-specified constraints on positions in 3D, positions of rulings, or folding angles. (Section 5.3.4)

5.3.2 Graph Constraints

On a developable surface, rulings intersect only at cone points; we thus require discrete folds to intersect each other only at (interior or boundary) vertices. Preventing the user from drawing folds that violate this constraint is straightforward, but preventing such intersections from occurring during optimization or simulation is much more difficult. Although non-intersection could be enforced with inequality constraints for each pair of folds, this approach requires up to $|F|^2$ nonlinear constraints for $|F|$ folds and quickly becomes prohibitively expensive.

We instead make two observations: first, we may assume, without loss of expressiveness, that the polygonal faces formed by cutting P along fold segments are convex. The folding angles of any folds around a concave interior vertex must be zero (it is impossible to fold along such lines without tearing the paper), so the surface is locally flat at that point and can be isometrically triangulated into convex pieces by inserting folds of angle zero. Second, given a graph in the plane with straight edges, whose faces are all convex, and whose edges do not intersect, it is impossible to smoothly slide vertices around in a way that introduces an edge-edge intersection without causing one of the faces to become non-convex.

We thus achieve non-intersection of edges by instead enforcing the simpler condition

that the *turning angle* between consecutive folds when traveling around each face of P lies in $(0, \pi)$. For polygons of winding number 1, this constraint is equivalent to convexity of the polygon. Polygons of higher winding number do not arise in practice, since it is impossible to arrive at such a configuration without violating the turning angle constraint of a neighboring face. See Figure 5.3 for examples of polygons satisfying and violating this constraint.

Since the boundary of a piece of paper does not grow or shrink during folding, we also cannot allow vertices on the boundary ∂P of the domain to leave ∂P. We enforce this condition by pinning the domain corners p_i, and restricting all other boundary vertices to slide along the lines between these corners.

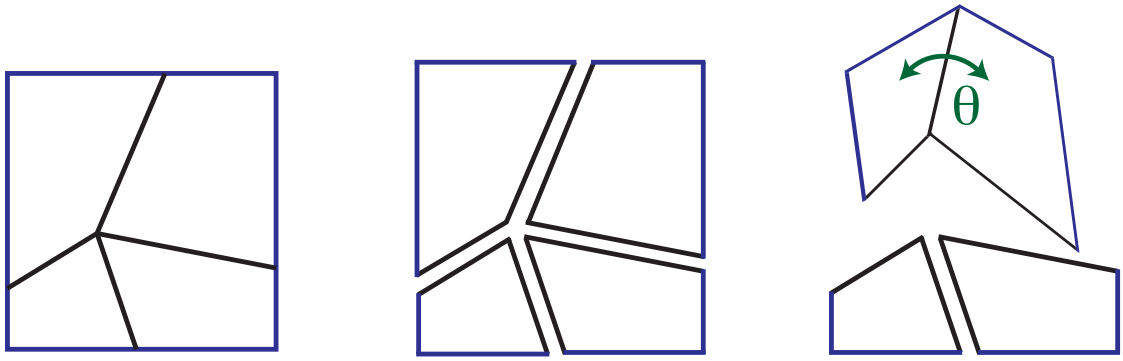
5.3.3 Compatibility Constraints

As long as folds do not intersect or meet (and thus all vertices are on the boundary), any configuration is admissible. For interior vertices, however, only some choices of folding angles for the adjacent fold lines yield a locally realizable surface.

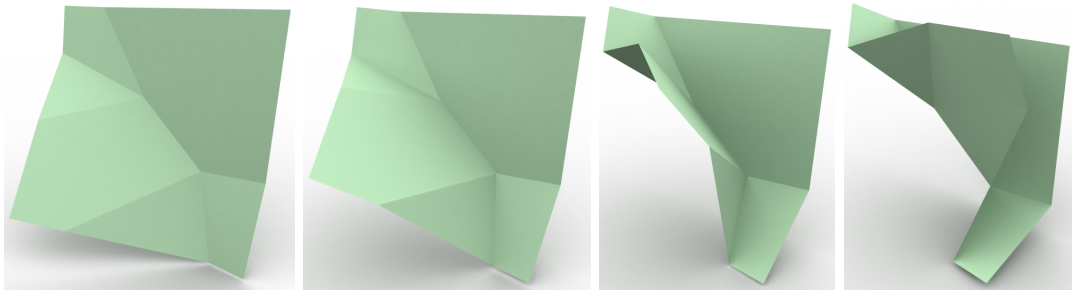
We can think of realizing a discrete developable surface as cutting P along its folds to form pieces (faces of P) to be assembled in $\mathbb{R}^3$ by gluing them together one at a time using angles in Θ, as illustrated in Figure 5.4(a). Often multiple such assemblies are possible with the same set of pieces, as in Figure 5.4(b). We must ensure that the pieces can be assembled in a way that does not tear the surface like the example in Figure 5.5. Unlike previous work on developable surface approximation using meshes, this constraint does *not* have to do with the requirement for vanishing Gaussian curvature, a property that follows automatically from our representation via rulings embedded on a flat sheet.

Consider the interior vertex $v \in V$ shown in Figure 5.6(a); without loss of generality we can assume that v is at position $(0, 0)$. We draw unit vectors $\hat{f}_1, \ldots, \hat{f}_k$ in the directions of the outgoing folds $f_1, \ldots, f_k$ from v in clockwise order. Embed P in the xy plane of $\mathbb{R}^3$ and define M_i to be the rotation about $\hat{f}_i$ by angle θ_i.

Suppose we cut outward from v along $\hat{f}_k$. We hold the piece of P counterclockwise from f_k flat on the xy plane and fold along $\hat{f}_1$ by angle θ_1, then along $\hat{f}_2$ by angle θ_2, and so on as in Figure 5.6(b). The product $M_1 M_2 \cdots M_i$ represents the transformation from



(a)



(b)

Figure 5.4: (a) Assembling a developable surface; (b) multiple sets of compatible folding angles are possible with a fixed set of points in V and ruling topology.

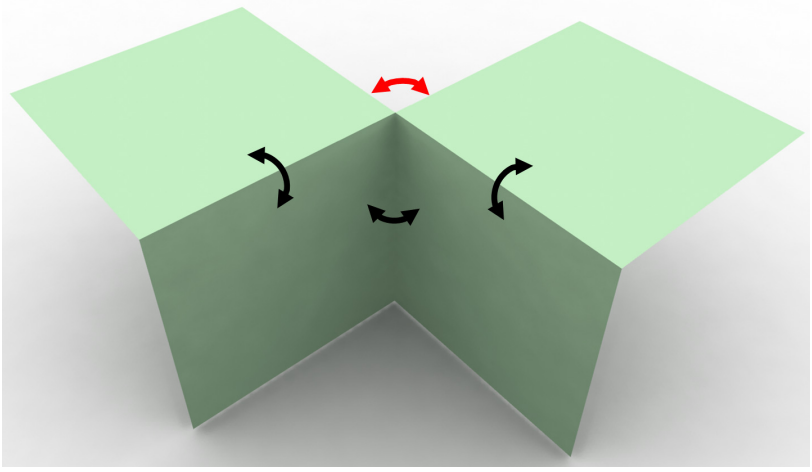


Figure 5.5: Prescribing the black folding angles in a simple crossed fold configuration forces a tear at the red angle, which should represent a single ruling.

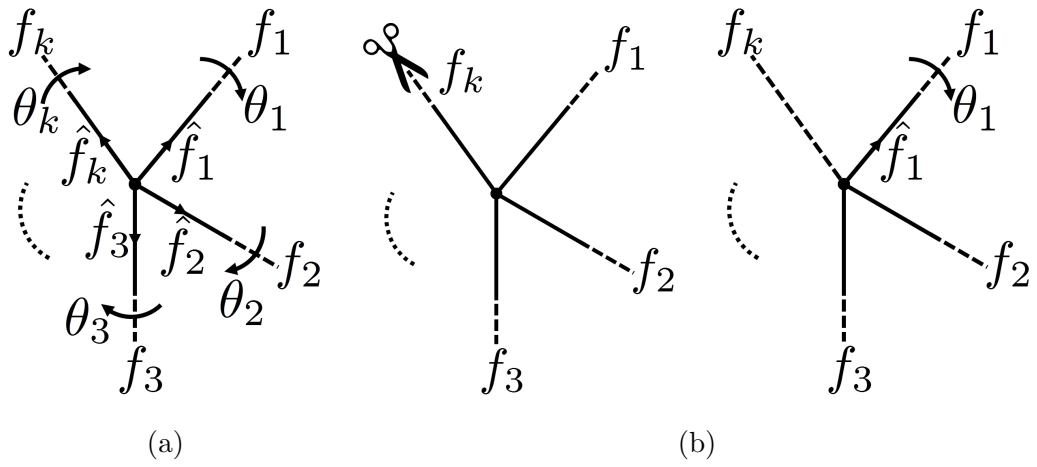


Figure 5.6: (a) Notation for interior vertex compatibility; (b) construction for compatibility constraint.

points in the segment clockwise from f_1 to f_i in the xy plane to its position in the three-dimensional configuration after i folds. Completing the loop, the product $M_1 M_2 \cdots M_k$ should put the segment clockwise from f_1 back in the xy plane, since we have completed the folding process and then rotated around the final fold using M_k. This process yields the compatibility condition $M_1 M_2 \cdots M_k = I_{3\times 3}$.

We should mention here that constraints in this form are redundant. In particular, the product $\prod_i M_i$ is guaranteed to be a rotation regardless of whether the configuration is compatible. It is thus sufficient to constrain only the three upper-off-diagonal entries to be zero. Strictly speaking, however, this constraint is weaker since it does not distinguish between the identity and 180° rotations about the axes, but this distinction is not important in practice for a surface that starts in a realizable configuration and deforms smoothly.

It is possible, but laborious, to construct a three-dimensional developable surface from any two-dimensional configuration satisfying all of the above constraints. To make it easier to realize the surface as it is being edited, and to enforce user-supplied constraints on the 3D positions of parts of the surface, we introduce as auxiliary degrees of freedom Euclidean transformations E mapping faces to their positions and orientations in $\mathbb{R}^3$. We then require additional compatibility conditions ensuring that these transformations do indeed reconstruct a seamless surface.

To guarantee that the transformations E realize the developable surface, it is sufficient

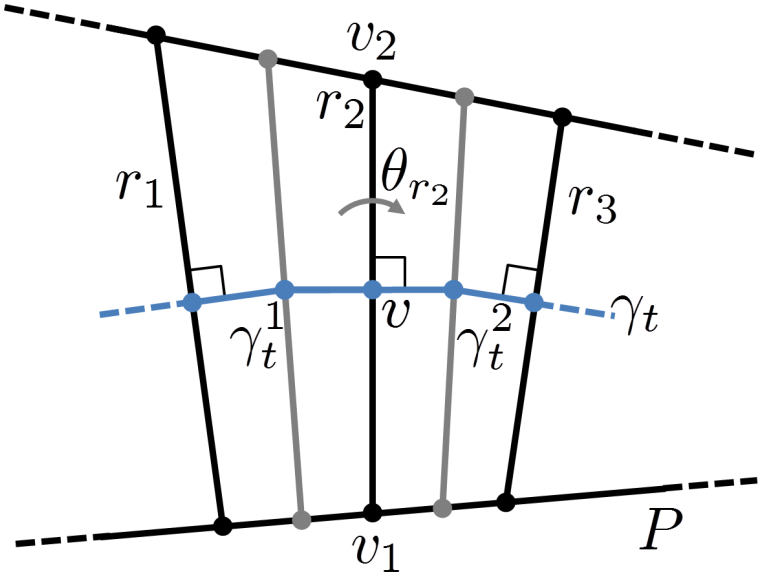


Figure 5.7: Notation for the mean curvature bending energy.

to build a minimum spanning tree of the dual graph of P and to enforce that two faces connected by an edge in the spanning tree abut in R^3, and with the correct folding angle. (That other pairs of adjacent faces meet up correctly is already enforced by the interior vertex compatibility constraints.) Thus, for each edge of our spanning tree we simply add constraints that endpoints of the adjacent pieces meet up and that the resulting fold is at the angle from Θ; the latter can be accomplished as a constraint on $\tan \frac{\theta_f}{2}$.

5.3.4 Additional Constraints

We allow the user to specify additional constraints on the configuration: vertex positions in P, folding angles, and the 3D positions of the vertices on the realization can all be pinned.

5.4 Mean Curvature Bending Energy

While the constructions in Section 5.3 can be used to obtain admissible developable surfaces, in the end we wish to approximate smooth surfaces rather than ones obtained solely using origami constructions. To this end, we introduce a squared mean curvature bending energy measuring a surface's deviation from flatness.

As in the smooth setting, we may infer curvature and thus bending energies using curves that run perpendicular to rulings. To this end, consider three rulings r_1, r_2, r_3 that border two neighboring planar pieces as in Figure 5.7. Assume the ends of r_2 are given by vertices v_1 and v_2. Then, for every point $v = (1-t)v_1 + tv_2$ on r_2, there exists a piecewise linear curve γ_t, comprised of three segments that are orthogonal to the rulings, whose vertices lie along the angle bisectors between r_1, r_2 and r_2, r_3. The bending energy induced by a folding angle θ_{r_2} about r_2 is associated with the part of the surface flanking either side of r_2 through the vertices of γ_t. Denote these vertices as γ_t^1 and γ_t^2.

The curve γ_t is a discrete principal curve of the surface $\forall t \in [0,1]$, because it is perpendicular to the zero-curvature ruling direction. Thus, we can regard θ as the mean curvature of our discrete developable integrated along the segment from γ_t^1 to γ_t^2. In particular, defining $s(t) = \|\gamma_t^2 - \gamma_t^1\|$ we write the *discrete mean curvature* of the surface at v as:

$$H(t) = \frac{\theta_{r_2}}{s(t)} \tag{5.1}$$

Of course, $H(t)$ is rarely constant as a function of t, representing the fact that the same folding angle θ_{r_2} gives different mean curvatures as the rulings become more and more spread out.

The curve γ_t can leave the boundary of the sheet. In this case we still take s to be the length of the segment of γ_t intersecting r_2 extended beyond the boundary of P; this way curvature does not shrink simply because P was cut to have a particular shape. Take $l(t)$ to be the length of γ_t between vertices adjacent to r_2 intersected with P; obviously we always have $l(t) \leq s(t)$.

Since our rulings and the border of the planar sheet are both comprised of line segments, $s(t)$ and $l(t)$ are piecewise linear functions of t. Thus, we can identify a sequence $t_0 = 0, t_1, \ldots, t_k = 1$ such that $s(t)$ and $l(t)$ are linear between adjacent values t_i; denote $s_i = s(t_i)$ and $l_i = l(t_i)$. We divide the computation of the bending energy associated with r_2 into computation of bending energies for each segment $[t_i, t_{i+1})$.

Taking into account this piecewise linear structure, for a given segment i, we can write

our bending energy using the following integral:

$$\begin{aligned} E_i &= d\Delta_t \int_0^1 \frac{\theta_{r_2}^2((1-u)l_i + ul_{i+1})}{((1-u)s_i + us_{i+1})^2}\, du \\ &= \frac{d\Delta_t\theta_{r_2}^2\left[\Delta_s(l_i s_{i+1} - l_{i+1}s_i) - \Delta_l s_i s_{i+1}\log\frac{s_{i+1}}{s_i})\right]}{s_i s_{i+1}\Delta_s^2} \end{aligned} \tag{5.2}$$

where $d = \|v_2 - v_1\|$ and $\Delta_\chi = \chi_{i+1} - \chi_i$ for any variable χ; assume $s_i \leq s_{i+1}$ and otherwise swap accordingly. The total mean curvature bending energy associated with r_2 is simply $E = \sum_{i=1}^{k-1} E_i$.

While the terms of (5.2) are fairly complex, they can be computed in closed form from the local geometry of the discrete developable. Our implementation Solomon *et al.* [2012] computes the derivatives using the open source automatic differentiation library FADBAD++.

5.4.1 Dealing with Singularities

General developable surfaces may contain singular points of unbounded mean curvature. While the mathematical objects may contain such points, their physical counterparts cannot. Evidence is found in the localized permanent scars formed when a sheet of paper is crumpled. Near the singular points, the physical sheet must stretch to remain smooth, and this strain causes permanent damage.

The thin plate elastic potential—the sum of the squared mean curvature bending energy and a membrane stretching energy—is bounded only for smooth surfaces Audoly and Pomeau [2009]. Surfaces with singular points are outside the space of minimizers of this potential. This phenomenon is most clear in the geometry of a circular cone, whose rulings all emanate from a point; approaching the tip of the cone, the radius of the circle vanishes, as does the radius of curvature.

These *d-cone singularities* have been studied Huffman [1976]; Ben Amar and Pomeau [1997]; Cerda and Mahadevan [2005]; Audoly and Pomeau [2009] and although the energy contribution from ridges can be significant Lobkovsky *et al.* [1995]; Lobkovsky and Witten [1996], stretching around the cone point contributes relatively little energy by comparison. Intuitively, because the discrepancy arises in an arbitrarily small neighborhood, it does not

affect the solution in a macroscopic sense. Our implementation therefore *implicitly* cuts a small hole around each singularity when evaluating the energy, by setting $t_0 = \varepsilon > 0$ and correspondingly interpolating s_0 and l_0. Although incorporating the full cone energy remains future work, the form of (5.2) indicates that our approximation will at least not have strong dependence on the choice of ϵ, since the divergence of E_0 occurs like $O(\log \varepsilon)$ as $s_0 = \varepsilon \to 0$.

5.4.1.1 Crease Energy

By contrast, the energy of the surface near creases must incorporate both bending and significant stretching contributions. Consider a crease ruling of length L bounding two planar regions of a developable surface, where the dihedral angle between these regions is $\bar{\theta}$ and the bending angle of the crease is $\theta = \pi - \bar{\theta}$. If we approximate the crease by a piece of cylinder of radius r, we see that the total energy of the crease goes to infinity in the limit of a perfect crease $r \to \infty$. The actual geometry of the crease involves some sagging, stretching, and bending of the material around the crease, as determined by balance of the stretching and bending energies. Solving for exact shape of the optimal surface around the crease is intractable, but Witten Witten [2006] used scaling arguments to argue that the membrane energy of the surface near the crease is approximately

$$E_{\text{crease}} \propto L^{1/3}\theta^{5/3}. \tag{5.3}$$

Incorporating this energy into a simulation of the dynamics of a crumpling surface introduces new challenges: the forces ∇E_{crease} arising from this energy diverge as $L \to 0$, destabilizing the simulation if an edge becomes too short. Moreover, this scaling law assumes that the surface is flat around a sufficiently large neighborhood of the crease ruling, an assumption that becomes invalid if a triangle becomes too acute. Overcoming these challenges is ongoing work.

5.5 Discrete Developable Surface Editing

We have implemented a straightforward interactive tool Solomon *et al.* [2012] for exploring the manifold of admissible discrete developable surfaces. The interface displays views of the

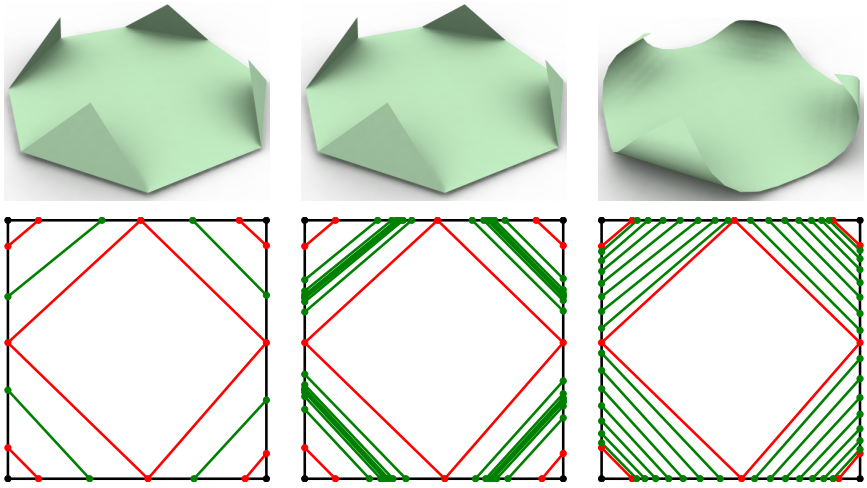
developable surface both in its two-dimensional configuration on P as well as the final three-dimensional shape. The user manually can introduce folds or rulings by drawing segments or snapping to the endpoints of existing folds; additional rulings can be introduced by the subdivision techniques described in Section 5.5.2. Constraints can be added by pinning vertices in the folded or unfolded configurations, fixing folding angles, and so on.

As the user attempts to change different aspects of the configuration in either of the two views, his or her motions are projected back onto the manifold of acceptable configurations. In particular, introduction of folds violating basic topological and geometric constraints explicitly is prohibited. More interestingly, when values like folding angles or vertex positions are changed, a iterative primal-dual interior point method with moderately tight error tolerances is used to find a nearby admissible configuration with respect to an energy measuring squared distance to the desired configuration. Since these projections occur for each small user-introduced change, few iterations are needed to return to a compatible configuration. Thus, our system can deal with moderately-sized ruling topologies including interior vertices interactively.

5.5.1 Curvature-based Relaxation

We can substitute the mean curvature bending energy (5.2) for the proximity measure from Section 5.5 while leaving the constraints as they are to formulate an energy minimization problem for relaxing rulings on discrete developable surfaces. Without additional constraints, this minimization simply will flatten the surface, since the lowest possible bending energy $E = 0$ is obtained when all fold angles are exactly zero. By pinning points in the folded configuration or by fixing even just one folding angle, however, the solution of this variational problem becomes far from trivial, attempting to smooth out the surface while moving only in the space of admissible developables.

We allow the user to specify where on the developable sheet the bending energy should be evaluated, distinguishing between ridges and rulings, the latter only appearing on smooth regions. Not all regions of the surface can be marked as smoothly folded, such as interior polygons within fold segments, which are provably flat; the user is restricted to smoothing those sections with or without cone singularities for which the bending energy can be eval-



(a) (b) (c)

Figure 5.8: The rough user-input configuration (a) is subdivided (b) to add rulings but preserve the geometry and then relaxed (c) using the mean curvature bending energy.

uated. While constraints can be added to localize the relaxation to a single portion of the developable surface, often times higher degrees of flexibility can be obtained by allowing more parts to move.

Figure 5.8 shows an example of curvature-based relaxation applied to a sharply-folded surface after applying the subdivision technique described below. As expected, this process smooths sharp corners while obeying user-specified constraints, providing a simple way to obtain a smooth developable from a rough configuration.

5.5.2 Subdivision

We provide a subdivision operator for refining smoothly-bent regions, similar in spirit to Liu et al. 2006b. We identify rulings whose folding angle is above some threshold $\hat{\theta}$ and add rulings with 0° folding angles flanking them on either side. We apply mean curvature relaxation to the subdivided surface and repeat until all fold angles are below $\hat{\theta}$. Combining subdivision, relaxation, and modeling yields an effective workflow for dealing with discrete developable surfaces, in which the user makes a rough model using few rulings as an initial step and then subdivides and relaxes to obtain the final surface, as exemplified in Figure 5.8.

We also provide a *crease* subdivision operator for approximating curved folds rather than

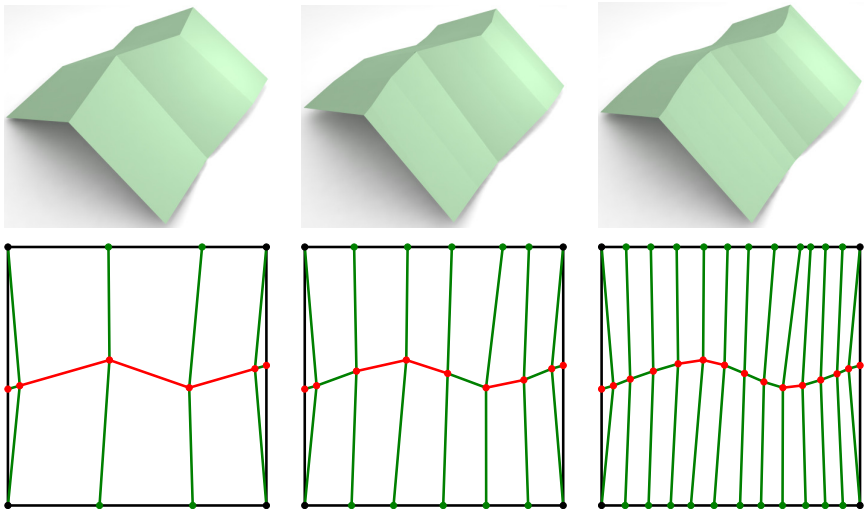


Figure 5.9: Successive application of our subdivision scheme generates smoother and smoother interior folds across the interior.

curved pieces. In this case, the user identifies chains of segments in $F \backslash R$ marked as belonging to a discrete curved fold. We apply four-point subdivision to the curve, yielding new vertices in the two-dimensional configuration. These vertices are connected using additional rulings to boundary segments on their neighboring faces, and the resulting configuration is relaxed. Two iterations of curved fold subdivision are shown in Figure 5.9. Note that the curved fold as well as the developable pieces on either side become smoother with each application of our operator.

5.5.3 Implementation Details

We use the IPOPT library for nonlinear optimization, with the Watson Sparse Matrix Package for linear algebra and the Eigen library for matrix operations. On a 2.4 GHz Xeon processor, we can deal with editing configurations with up to 50 rulings at interactive rates; better speeds likely could be achieved by making use of parallel computations to evaluate derivatives and project.

One aspect of our formulation that requires special attention during optimization is the enforcement of the fold-fold interaction constraints from Section 5.3.2. These constraints always guarantee admissibility but prevent folds from moving from one edge of P to another.

To circumvent this issue, during optimization we simply check for fold vertices coincident with vertices of P; if their energy gradient directs them into a different segment of P, we make that localized change to the topology of the surface and continue with optimization. Such a method can be regarded as an "active set" technique leaving constraints that folds do not leave far-away parts of P in the inactive set.

5.6 Alternative Degrees of Freedom

One of the main drawbacks of the above formulation are the highly nonlinear constraints. The compatibility constraints (5.3.3) around each interior vertex are compositions of many rotations that depend nonlinearly on the incident edges and turning angles. Moreover, since the degrees of freedom are associated to the material only, with the embedding dependent on these degrees of freedom, *external constraints* become difficult to enforce: even simple constraints like pinning a couple of points of the material to points in space amount to constraints that are long chains of function compositions coupling many degrees of freedom along the graph.

The use of the Euclidean motions E as extra, coupled degrees of freedom greatly simplify the latter constraints, and this idea can be extended further: we can specify that the degrees of freedom are simply the planar positions of the vertices v_i and their embedded positions q_i. The geometric constraints on the graph (section 5.3.2) remain, but the compatibility around each interior vertex need no longer be explicitly enforced; instead, we

- Triangulate the domain, introducing null (zero-angle) rulings as needed;
- Enforce *isometry constraints* $\|v_j - v_i\| = \|q_j - q_i\|$ for each pair of vertices i, j connected by a fold.

The benefits of this representation include simpler constraints (both the isometry and graph constraints are quadratic functions of the position DOFs) and the ability to render and manipulate the embedding of the surface without a costly reconstruction step. On the other hand, despite the simplicity of each individual constraint function, the constraints' feasible set is still non-convex, and we've observed that even for simple arrangements of

rulings and creases, such as the Yoshimura pattern, the feasible set has complex geometry with non-manifold features. Moreover, by introducing artificial rulings, triangulating P precludes simple ruling subdivision as described in section 5.5.2, and sophisticated sliding and remeshing of the rulings becomes necessary to avoid locking.

5.6.1 Numerical Challenges

As a preliminary experiment, we simulated crushing of Yoshimura patterns using the above DOFs and constraints: we generated a cylinder, discretized as a diamond pattern, and embedded it so that its axis of symmetry was aligned to the z axis. We added additional constraints fixing one end cap; the vertices of the second end cap were constrained to lie on a plane parallel to and at distance h from the fixed end cap (in other words, the second end cap remained free to rotate about the z axis).

We the quasistatically crushed the cylinder: we slowly decreased h; after every decrease, we attempted to solve, using black-box interior point methodsWächter and Biegler [2006], for a configuration that satisfied all of the isometry and boundary constraints and minimized the total bending energy of the surface. Although this optimization problem converges for small amounts of axial compression, we have not yet successfully crushed the cylinder fully. Several numerical challenges arose, and overcoming these remains promising future work:

Rank-deficiency of the constraints The isometry constraints are not typically algebraically independent. For instance, the discrete Gauss-Bonnet theorem couples the boundary and isometry constraints. Moreover, the number of independent isometry constraints cannot exceed the number of interior vertices, since an alternative formulation of global isometry is that all of the vertices of the embedding have vanishing discrete Gaussian curvature. Since a regular triangular mesh has roughly three times as many edges as vertices (depending on the configuration of the boundary), only about one-third of the isometry constraints are independent. Lastly, the number of independent constraints can change as the cylinder is compressed: we observe that at symmetrical configurations of the embedded cylinder (such as the Yoshimura pattern initial conditions), the number of indepedent constraints is significantly lower than for general configurations.

The above implies that the feasible region of configuration space, where all of the constraints are satisfied, is not globally manifold; the optimization algorithms we have tried do not robustly handle singularities where the dimension changes. A better understanding of how to construct a *minimal* set of isometry constraints, or use of algorithms like continuation methods that are designed to handle these singularities, are possible future solutions.

Many critical points Unsurprisingly, for a given value of h there are many feasible configurations which are critical points of the bending energy, besides the global minimum. For small amounts of compression, symmetric configurations minimize the objective; however after some critical amount of compression these configurations become saddle points, and are replaced by many local minima that are very asymmetric and distant from the symmetric configuration. This symmetry-breaking and bifurcation is expected, but difficult to simulate robustly.

Collapsing edges The crease energy described in section 5.4.1.1 scales as $L^{1/3}$, where L is the crease length; thus its derivative scales as $L^{-2/3}$, which increases as L decreases. This scaling behavior creates a positive feedback, where shortening already short edges is highly energetically favorable, and has three negative effects on numerical simulation. First, edges are often driven to zero length during optimization, and unfortunately the objective function, which depends on mesh edge lengths and angles, is poorly conditioned when edge lengths vanish; second, the crease energy can have minima at points where the energy is not differentiable; and lastly, the scaling arguments Witten [2006] underlying the expression for the crease energy break down for creases that become too short relative to their neighbors. We experimented with solving the first two problems by introducing a *merging* step into the optimization, where edges are collapsed once their length falls below a threshold; perhaps another approach, revisiting the arguments of Witten and taking into account the more complex geometry at the intersection of several creases, could solve all three.

Adding vertices and edges To correctly reproduce the crushing of a cylinder, we need to allow not only merging and rearrangement of the existing vertices, put also insertion of

new degrees of freedom. Possible simple heuristics include subdividing triangles when the stress passing through them, as estimated using the Lagrange multipliers of the isometry constraints, exceeds a threshold. However, edge insertion must be done carefully, since the crease energy is only valid for creases that are not too close to each other. For instance, since the crease energy scales as $L^{1/3}\theta^{5/3}$, splitting the vertices at the ends of an edge and adding a new, coincident edge artificially *decreases* the energy of that edge:

$$2\frac{L^{1/3}\theta^{5/3}}{2^{5/3}} \leq L^{1/3}\theta^{5/3}.$$

A careful investigation of how to adaptively add new degrees of freedom to the simulation remains future work.

5.7 Conclusion and Discussion

The method described above and in my earlier work Solomon *et al.* [2012] takes some initial steps towards simulating the behavior of thin elastic sheets, in the stess-localized regime, using discrete developable surfaces. Developable surfaces, approximated as a network of rulings and ridge lines, are a natural reduced representation of such sheets – by taking advantage of the extra geometric structure we know is present in the deformation of thin elastic sheets (isometry away from a sparse set of singular lines and points), modeling this deformation with discrete developable surfaces promises to efficiently capture the right bending and crumpling behavior without needing to resolve the computationally-expensive stretching modes.

Fully realizing this goal will require long-term research in several directions:

- A more complete understanding of when an elastic sheet can be approximated by a piecewise developable surface: for a buckled elastic sheet, is it always possible to partition the sheet into regions of localized and diffuse stress Schroll *et al.* [2011]? Can these regions be predicted from the loads and boundary conditions?

- Is there an efficient, reduced representation for surfaces with *nearly* zero Gaussian curvature? Perhaps earlier work on noncomforming plate elements Botsch *et al.* [2006];

English and Bridson [2008]; Mitra *et al.* [2008] or particle-based methods such as is used in peridynamic simulations S.A. [2000] could prove useful.

- For regions with localized stress only, the crease energy is appropriate for approximating the membrane energy due to local stretching of the surface near the crease, but only for creases that are sufficiently long and sufficiently far apart from other creases. Is the crease energy appropriate for simulating the transition between crease patterns that occurs e.g. when a metal cylinder is crushed?

- What is the role of plasticity in the crushing behavior of materials like paper and aluminum? How can this plasticity be incorporated into a model of developable elastic sheets?

Part III

Conclusions

Chapter 6

Conclusions

The laws of physics governing the world around us are written in the language of geometry. To take full advantage of computation to study physical systems, we must write computer programs that also speak that language; unfortunately, standard numerical techniques are often blind to, or worse, outright destroy the geometric structure at the heart of these physical systems. Thin, surface-like objects, including pieces of cloth, freeform masonry and steel-glass structures, and sheets of paper or sheet metal, are particularly rich in geometric structure, and the projects I've outlined in this thesis successfully took advantage of that structure to develop algorithms for studying these objects in a way that is more efficient and more accurate than previously possible.

For the first time, geometric integration allows us to simulate the contact of cloth with itself in a way that guarantees that no collisions are missed, that momentum and energy does not drift over time, and that the algorithm terminates for all well-posed input. Our understanding of the geometry of time-integration and contact allowed us to dramatically improve the performance of this algorithm without damaging any of these guarantees. Understanding that the stability of masonry structures is a function of its shape, and not its material properties, allowed us to analyze and design freeform masonry structures more efficiently, and design new algorithms for manipulating such surfaces, for instance taking any stable surface and building it out of planar quadrilateral panels. Thin elastic sheets often exhibit stress-focusing when buckling, and such sheets can be approximated in a geometry-aware way using discrete developable surfaces. We built a tool for designing developable

shapes using this representation, and similar ideas promise to allow us to fully simulate buckling surface in the future.

I've sketched possible extensions and directions of future research for many of these specific problems in the preceding chapters; in addition to these immediate directions, I believe there exist several broad areas related to the geometry-aware discretization of physical systems where significant future research remains to be done. Solving these problems would take us large steps towards fully understanding how to best leverage our computational resources to predict and control the world around us.

Limits of Discretization Unfortunately it is not always possible to simultaneously translate all of a smooth problem's geometric structure into discrete structure. For example, it is known Zhong and Marsden [1988] that it is impossible to simulate the solar system in a way that exactly preserves its momentum, energy, and so-called *symplectic form* (a similar geometric invariant). The Laplacian, a ubiquitous differential operator central to the equations governing the flow of heat or propagation of waves, cannot be discretized in a way that preserves all of the smooth Laplacian's potentially desirable properties Wardetzky *et al.* [2007]. *Understanding the limitations* of the discrete theory prevents wasting time looking for a "perfect" discretization where none exists, and makes explicit what tradeoffs must be made when weighing different discretizations for use in a particular application. Moreover, once the limitations are known, we can look for loopholes: for instance, as described in chapter 2, it is possible for some physical systems (elastic shells undergoing contact) to exactly preserve momentum and the symplectic form while keeping the energy error *bounded* for exponentially long time.

Unified Handling of Constraints In most interesting physical systems, objects aren't free to move arbitrary, but are restricted by constraints. Chapter 2 discussed one common type of constraint, that two objects cannot pass through each other. Various types of kinematic constraints are also common: my upper and lower arm are free to swivel with respect to each other, but remain attached at the elbow. Cloth can stretch slightly, but becomes inextensible after a point. Thin elastic surfaces, at least in some cases, deform without undergoing intrinsic stretching (chapter 5).

The geometry of smooth constrained mechanics has been thoroughly studied, but there does not yet exist a satisfactory discretization of this geometry. Current approaches are either computationally expensive, do not guarantee that the constraints are always obeyed, or do not respect the system's invariants like conservation of energy. Exploring *discretizations of the constraint geometry* itself would lead to explicit, efficient, and correct algorithms for simulating these common physical systems.

Inverse Problems My work on stable masonry surfaces (chapter 4) includes an example of an *inverse* or design problem: instead of asking how a given structure reacts to some imposed forces (the forward problem), the inverse problem asks, for a given user-specified target shape, whether there exists a nearby stable surface. The inverse problem tends to be more challenging than the forward problem, but also rich in applications. For example, the growth of many biological tissues can be modeled as two thin layers growing at different rates, causing the surface to buckle. The ability to design a seed surface given a target shape would be important for tissue engineering and the manufacture of structures made from biomimetic materials. A solid understanding of the problem's geometry is critical to successfully solving these inverse problems: first, to guarantee that the chosen discretization of the problem will find an approximate solution if a smooth solution exists, and vice versa; and second, to provide insights into what properties a solution must possess, as footholds for finding one.

Part IV

Bibliography

Bibliography

S. Ainsley, E. Vouga, E. Grinspun, and R. Tamstorf. Speculative parallel asynchronous contact mechanics. *ACM Trans. Graph.*, 31(6):151:1–151:8, November 2012.

M. B. Amar and Y. Pomeau. Crumpled paper. *Proc. R. Soc. Lond.*, 453:729–755, 1997.

A. Andreu, L. Gil, and P. Roca. Computational analysis of masonry structures with a funicular model. *J. Engrg. Mechanics*, 133(473-480), 2007.

P. Ash, E. Bolker, H. Crapo, and W. Whiteley. Convex polyhedra, Dirichlet tessellations, and spider webs. In *Shaping space (Northampton 1984)*, pages 231–250. Birkhäuser, 1988.

B. Audoly and Y. Pomeau. *Elasticity and Geometry: From hair cblahurls to the nonlinear response of shells*. Oxford University Press, USA, December 2009.

G. Aumann. A simple algorithm for designing developable Bézier surfaces. *Computer Aided Geometric Design*, 20(8-9):601–619, 2003.

G. Aumann. Degree elevation and developable Bézier surfaces. *Computer Aided Geometric Design*, 21(7):661–670, 2004.

M. R. Barnes. Form finding and analysis of tension structures by dynamic relaxation. *Int. J. Space Struct.*, 14(2):89–104, 2009.

E. Barth, B. Leimkuhler, and S. Reich. A time-reversible variable-stepsize integrator for constrained dynamics. *SIAM Journal on Scientific Computing*, 21(3):1027–1044, 1999.

T. Belytschko and M. O. Neal. Contact-impact by the pinball algorithm with penalty

and lagrangian methods. *International Journal for Numerical Methods in Engineering*, 31(3):547–572, 1991.

T. Belytschko, W. J. T. Daniel, and G. Ventura. A monolithic smoothing-gap algorithm for contact-impact based on the signed distance function. *International Journal for Numerical Methods in Engineering*, 55(1):101–125, 2002.

T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley and Sons, 2006.

M. Ben Amar and Y. Pomeau. Crumpled paper. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 453(1959):729–755, April 1997.

J. Bender and D. Bayer. Parallel simulation of inextensible cloth. In *Proceedings of VRIPhys*, 2008.

D. P. Bertsekas. *Contrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1984.

R. L. Bishop. There is More than One Way to Frame a Curve. *American Mathematical Monthly*, 82(3):246–251, 1975.

D. L. Blair and A. Kudrolli. The geometry of crumpled paper. *Phys. Rev. Lett.*, 94, 2005.

P. Block and L. Lachauer. Closest-fit, compression-only solutions for free form shells. In *IABSE — IASS 2011 London Symposium*. Int. Ass. Shell Spatial Structures, 2011. [electronic].

P. Block and J. Ochsendorf. Thrust network analysis: A new methodology for three-dimensional equilibrium. *J. Int. Assoc. Shell and Spatial Structures*, 48(3):167–173, 2007.

P. Block. *Thrust Network Analysis: Exploring Three-dimensional Equilibrium*. PhD thesis, Massachusetts Institute of Technology, 2009.

P. Bo and W. Wang. Geodesic-controlled developable surfaces for modeling paper bending. *Computer Graphics Forum*, 26(3):365–374, 2007.

A. Bobenko, P. Schröder, J. Sullivan, and G. Ziegler, editors. *Discrete Differential Geometry*. Birkhäuser, 2008.

A. Bobenko, H. Pottmann, and J. Wallner. A curvature theory for discrete surfaces based on mesh parallelity. *Math. Annalen*, 348:1–24, 2010.

M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

A. Boudaoud, P. Patricio, Y. Couder, and M. B. Amar. Dynamics of singularities in a constrained elastic plate. *Letters to Nature*, 407, 2000.

R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact, and friction for cloth animation. *ACM Transactions on Graphics*, 21(3):594–603, 2002.

R. Bridson. *Computational aspects of dynamic surfaces*. PhD thesis, Stanford University, 2003.

T. Brochu, E. Edwards, and R. Bridson. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics (TOG) – Proceedings of the ACM SIGGRAPH 2012*, 2012.

R. Burgoon, E. Grinspun, and Z. Wood. Discrete Shells Origami. In *Proceedings of Computers And Their Applications*, pages 180–187, March 2006.

M. P. D. Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

E. Cerda and L. Mahadevan. Conical surfaces and crescent singularities in crumpled sheets. *Physical Review Letters*, 80(11), 1998.

E. Cerda and L. Mahadevan. Geometry and physics of wrinkling. *Physical Review Letters*, 90(7), 2003.

E. Cerda and L. Mahadevan. Confined developable elastic surfaces: cylinders, cones and the Elastica. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2055):671–700, March 2005.

D. Chapelle and K. J. Bathe. Fundamental considerations for the finite element analysis of shell structures. *Computers and Structures*, 66:19–36, 1998.

M. Chen and K. Tang. A fully geometric approach for developable cloth deformation simulation. *Vis. Comput.*, 26:853–863, 2010.

M. Chen and K. Tang. Quasi-developable surface modeling of contours with curved triangular patches. *Computers and Graphics*, 2013.

H.-Y. Y. Chen, I.-K. K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On Surface Approximation Using Developable Surfaces. *Graphical Models and Image Processing*, 61(2):110–124, March 1999.

C.-H. Chu and C. H. Squin. Developable bzier patches: properties and design. *Computer-Aided Design*, 34(7):511 – 527, 2002.

F. Cirak and M. West. Decomposition contact response (DCR) for explicit finite element dynamics. *International Journal for Numerical Methods in Engineering*, 64(8):1078–1110, 2005.

D. Cohen-Steiner and J.-M. Morvan. Restricted Delaunay triangulations and normal cycle. In *Proc. 19th Symp. Computational Geometry*, pages 312–321. ACM, 2003.

E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *ACM '69 Proceedings of the 1969 24th national conference*, pages 157–172, 1969.

F. de Goes, P. Alliez, H. Owhadi, and M. Desbrun. On the equilibrium of simplicial masonry structures. *ACM Transactions on Graphics*, 2013.

G. Debunne, M. Desbrun, M.-P. Cani, and A. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Proceedings of SIGGRAPH 01*, pages 31–36, 2001.

B. A. DiDonna. Scaling of the buckling transition of ridges in thin sheets. *Physical Review E*, 66, 2002.

J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodriguez-Ferran. *Encyclopedia of Computational Mechanics*, chapter Arbitrary Lagrangian-Eulerian Methods. John Wiley & Sons, 2004.

E. English and R. Bridson. Animating developable surfaces using nonconforming elements. *ACM Transactions on Graphics*, 27, 2008.

F. Fraternali, M. Angelillo, and A. Fortunato. A lumped stress method for plane elastic problems and the discrete-continuum approximation. *Int. J. Solids Struct.*, 39:6211–6240, 2002.

F. Fraternali. A thrust network approach to the equilibrium problem of unreinforced masonry vaults via polyhedral stress functions. *Mechanics Res. Comm.*, 37(2):198 – 204, 2010.

W. Frey. Boundary triangulations approximating developable surfaces that interpolate a closed space curve. *International Journal of Foundations of Computer science*, 13(2002):285–302, 2002.

W. Frey. Modeling buckled developable surfaces by triangulation. *Computer Aided Design*, 36:299–313, 2004.

M. P. Friedlander. BCLS: Bound constrained least squares. *http://www.cs.ubc.ca/∼mpf/bcls*, 2007.

M. Giaquinta and E. Giusti. Researches on the equilibrium of masonry structures. *Archive for Rational Mechanics and Analysis*, 88(4):359–392, 1985.

A. Green and W. Zerna. *Theoretical Elasticity*. Dover, 2002. Reprint of the 1968 edition.

L. Gründig, E. Moncrieff, P. Singer, and D. Ströbel. A history of the principal developments and applications of the force density method in Germany 1970–1999. In *4th Int. Coll. Computation of Shell & Spatial Structures*, 2000.

L. J. Guibas. Kinetic data structures: a state of the art report. In *Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics on Robotics : the Algorithmic Perspective*, pages 191–209, 1998.

E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer-Verlag, second edition, 2006.

D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun. Asynchronous contact mechanics. In *ACM SIGGRAPH 2009 papers*, pages 1–12, 2009.

D. Harmon, Q. Zhou, and D. Zorin. Asynchronous integration with phantom meshes. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2011.

D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun. Asynchronous Contact Mechanics (CACM Research Highlights). *Communications of the ACM*, 55(4):102—-109, Apr 2012.

D. Harmon. *Robust, Efficient, and Accurate Contact Algorithms*. PhD thesis, Columbia University, 2010.

H. Hertz. Study on the contact of elastic bodies. *Journal für die Reine und Angewandthe Mathematik*, 29:156–171, 1882.

J. Heyman. The stone skeleton. *Int. J. Solids Structures*, 2:249–279, 1966.

J. Heyman. *The Stone Skeleton: Structural Engineering of Masonry Architecture*. Cambridge University Press, 1995.

J. Heyman. *Structural Analysis: A Historical Approach*. Cambridge University Press, 1998.

N. J. Hoff, W. A. Madsen, and J. Mayers. Postbuckling equilibrium of axially compressed circular cylindrical shells. *AIAA Journal*, 4(1), 1966.

J.-C. Huang, X. Jiao, R. M. Fujimoto, and H. Zha. DAG-guided parallel asynchronous variational integrators with super-elements. In *Proceedings of the 2007 summer computer simulation conference*, pages 691–697, 2007.

D. A. Huffman. Curvature and creases: A primer on paper. *Computers, IEEE Transactions on*, 100(10):1010–1019, 1976.

T. Hughes, R. L. Taylor, J. L. Sackman, A. Curnier, and W. Kanoknukulchai. A finite element method for a class of contact-impact problems. *Computer Methods in Applied Mechanics and Engineering*, 8:249–276, 1976.

T. Ida and H. Takahashi. Origami fold as algebraic graph rewriting. *Journal of Symbolic Computation*, 45(4):393–413, April 2010.

T. Ida. Graph Rewriting in Computational Origami. In *2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, volume 1, pages 20–27. IEEE, 2008.

D. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7:404–425, 1985.

D. Julius, V. Kraevoy, and A. Sheffer. D-Charts: Quasi-Developable Mesh Segmentation. *Computer Graphics Forum*, 24(3):581–590, 2005.

K. Kale and A. Lew. Parallel asynchronous variational integrators. *International Journal for Numerical Methods in Engineering*, 70:291–321, 2007.

C. Kane, E. A. Repetto, M. Ortiz, and J. E. Marsden. Finite element analysis of nonsmooth contact. *Computer Methods in Applied Mechanics and Engineering*, 180(1):1–26, 1999.

C. Kane, J. E. Marsden, M. Ortiz, and M. West. Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems. *International Journal for Numerical Methods in Engineering*, 49(10):1295–1325, 2000.

Y. Kergosien, H. Gotoda, and T. Kunii. Bending and creasing virtual paper. *IEEE Computer Graphics and Applications*, 1994.

N. Kikuchi and J. T. Oden. *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite element Methods*. SIAM, 1988.

A. Kilian and J. Ochsendorf. Particle-spring systems for structural form finding. *J. Int. Assoc. Shell and Spatial Structures*, 46:77–84, 2005.

M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann. Curved folding. *ACM Transactions on Graphics*, 27(3):1, August 2008.

D. Kim, J.-P. Heo, J. Huh, J. Kim, and S.-E. Yoon. HPCCD: Hybrid parallel continuous collision detection using cpus and gpus. *Computer Graphics Forum (Pacific Graphics)*, 28(7), 2009.

J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

P. Konečný and K. Zikan. Lower bound of distance in 3D. In *Proceedings of WSCG 1997*, volume 3, pages 640–649, 1997.

T. Kotnik and M. Weinstock. Material, form and force. *Architectural Design*, 82:104–111, 2012.

S. K. Lahiri, J. Bonet, and J. Peraire. A variationally consistent mesh adaptation method for triangular elements in explicit lagrangian dynamics. *International Journal for Numerical Methods in Engineering*, pages 1–39, 2008.

C. Lanczos. *The Variational Principles of Mechanics*. Dover Publications, fourth edition, 1986.

L. D. Landau, L. P. Pitaevskii, E. M. Lifshitz, and A. M. Kosevich. *Theory of Elasticity*. Butterworth-Heineman, 1986.

T. A. Laursen and G. R. Love. Improved implicit integrators for transient impact problems - geometric admissibility within the conserving framework. *International Journal for Numerical Methods in Engineering*, 53(2):245–274, 2002.

T. Laursen. *Computational Contact and Impact Mechanics*. Springer, 2002.

S. Leopoldseder and H. Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design*, 30(7):571–582, 1998.

A. Lew, J. Marsden, M. Ortiz, and M. West. Asynchronous variational integrators. *Archive for Rational Mechanics and Analysis*, 167(2):85–146, 2003.

A. Lew. *Variational Time Integrators in Computational Solid Mechanics*. PhD thesis, California Institute of Technology, 2003.

K. Linkwitz and H.-J. Schek. Einige Bemerkungen zur Berechnung von vorgespannten Seilnetzkonstruktionen. *Ingenieur-Archiv*, 40:145–158, 1971.

Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graphics*, 25(3):681–689, 2006. Proc. SIGGRAPH.

Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Transactions on Graphics*, 25(3):681, July 2006.

Y.-J. Liu, K. Tang, and A. Joneja. Modeling dynamic developable meshes by the Hamilton principle. *Computer-aided Design*, 39:719–731, 2007.

Y.-J. Liu, K. Tang, and A. Joneja. Modeling dynamic developable meshes by the Hamilton principle. *Computer-Aided Design*, 39(9):719–731, September 2007.

Y. Liu, P. Hao, J. Snyder, W. Wang, and B. Guo. Computing self-supporting surfaces by regular triangulation. *ACM Transactions on Graphics*, 2013.

R. K. Livesley. A computational model for the limit analysis of three-dimensional masonry structures. *Meccanica*, 27:161–172, 1992.

A. Lobkovsky and T. A. Witten. Properties of ridges in elastic membranes. *Physical Review E*, 55:1577–1589, 1996.

A. Lobkovsky, S. Gentges, H. Li, D. Morse, and T. A. Witten. Scaling properties of stretching ridges in a crumpled elastic sheet. *Science*, 270(5241):1482–1485, 1995.

R. MacKay. Some aspects of the dynamics of hamiltonian systems. In *The Dynamics of Numerics and the Numerics of Dynamics*, pages 137–193. Oxford University Press, 1992.

L. Mahadevan, A. Vaziri, and M. Das. Persistence of a pinch in a pipe. *Europhysics Letters*, 77(4), 2007.

J. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, 2001.

J. Maxwell. On reciprocal diagrams and diagrams of forces. *Philosophical Magazine*, 4(27):250–261, 1864.

B. Mirtich. Timewarp rigid body simulation. *SIGGRAPH '00 Proceedings of the 27th annual conference of computer graphics and interactive techniques*, pages 193–200, 2000.

J. Mitani and H. Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics*, 23(3):259, August 2004.

N. Mitra, M. Kilian, S. Floery, Z. Chen, A. Sheffer, and H. Pottmann. Developable Surfaces with Curved Creases. *Advances in Architectural Geometry*, 2008.

J. Moser and A. P. Veselov. Discrete versions of some classical integrable systems and factorization of matrix polynomials. *Communications in Mathematical Physics*, 139(2):217–243, 1991.

J. Mosler and M. Ortiz. On the numerical implementation of variational arbitrary lagrangian-eulerian (VALE) formulations. *International Journal for Numerical Methods in Engineering*, 2006.

P. Mullen, P. Memari, F. de Goes, and M. Desbrun. HOT: Hodge-optimized triangulations. *ACM Transactions on Graphics (TOG): Proceedings of SIGGRAPH 2011*, 2011.

R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6):152:1–152:10, November 2012.

R. Narain, T. Pfaff, and J. F. O'Brien. Folding and crumpling adaptive sheets. *ACM Trans. Graph.*, 32(4):51:1–51:8, July 2013.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2000.

B. Nour-Omid and P. Wriggers. A two-level iteration method for solution of contact problems. *Computer Methods in Applied Mechanics and Engineering*, 54:131–144, 1986.

J. T. Oden. Exterior penalty methods for contact problems in elasticity. In Bathe, Stein, and Wunderlich, editors, *Europe-US Workshop: Nonlinear Finite Element Analysis in Structural Mechanics*. Springer, 1980.

D. O'Dwyer. Funicular analysis of masonry vaults. *Computers and Structures*, 73:187–197, 1998.

S. Pabst, A. Koch, and W. Straer. Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. *Computer Graphics Forum*, 29(5):1605–1612, 2010.

A. Pandolfi, C. Kane, J. E. Marsden, and M. Ortiz. Time-discretized variational formulations of non-smooth frictional contact. *International Journal for Numerical Methods in Engineering*, 53(8):1801–1829, 2002.

D. Panozzo, P. Block, and O. Sokine-Hornung. Designing unreinforced masonry models. *ACM Transactions on Graphics*, 2013.

F. Pérez and J. Suárez. Quasi-developable -spline surfaces in ship hull design. *Computer-Aided Design*, 39(10):853–862, October 2007.

D. Perić and D. R. J. Owen. Computational model for 3-D contact problems with friction based on the penalty method. *International Journal for Numerical Methods in Engineering*, 35(6):1289–1309, 1992.

M. Perriollat. A quasi-minimal model for paper-like surfaces. *Computer Vision and Pattern Recognition*, pages 1–7, 2007.

K. Pingali, D. Nguyen, M. Kulkarni, M. Burtscher, M. A. Hassaan, R. Kaleem, T.-H. Lee, A. Lenharth, R. Manevich, M. Méndez-Lojo, D. Prountzos, and X. Sui. The tao of parallelism in algorithms. In *Proceedings of the 32nd ACM SIGPLAN conference on programming language design and implementation*, pages 12–25, 2011.

H. Pottmann and Y. Liu. Discrete surfaces in isotropic geometry. In M. Sabin and J. Winkler, editors, *Mathematics of Surfaces XII*, pages 341–363. Springer, 2007.

H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer, 2010.

H. Pottmann, Y. Liu, J. Wallner, A. Bobenko, and W. Wang. Geometry of multi-layer freeform structures for architecture. *ACM Trans. Graphics*, 26(3):#65,1–11, 2007. Proc. SIGGRAPH.

R. Rangarajan, R. Ryckman, and A. Lew. Towards long-time simulation of soft tissue simulant penetration. In *Proceedings of the 26th Army Science Conference*, pages 1–8, 2008.

P. Redont. Representation and deformation of developable surfaces. *Computer-Aided Design*, 21(1):13–20, 1989.

D. Rohmer, M.-P. Cani, S. Hahmann, and B. Thibert. Folded paper geometry from 2D pattern and 3D contour. *Eurographics 2011*, pages 21–24, 2011.

K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert. Developable surfaces from arbitrary sketched boundaries. *Proc. Symposium on Geo. Processing*, pages 163–172, 2007.

R. Ryckman and A. Lew. An explicit asynchronous contact algorithm for elastic body-rigid wall interaction. *International Journal for Numerical Methods in Engineering*, 89:869–896, 2011.

S. S.A. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids*, 48(1):175–209, 2000.

M. Schenk and S. D. Guest. Origami Folding: A Structural Engineering Approach. *Origami 5 Fifth International Meeting of Origami Science Mathematics and Education*, pages 1–16, 2011.

A. Schiftner and J. Balzer. Statics-sensitive layout of planar quadrilateral meshes. In C. Ceccato et al., editors, *Advances in Architectural Geometry 2010*, pages 221–236. Springer, Vienna, 2010.

A. Schiftner. Planar quad meshes from relative principal curvature lines. Master's thesis, TU Wien, 2007. *http://dmg.tuwien.ac.at/aschiftner/doc/diplomarbeit.pdf*.

R. Schroll, E. Katifori, and B. Davidovich. Elastic building blocks for confined sheets. *Physical Review Letters*, 106, 2011.

A. Selle, J. Su, G. Irving, and R. Fedkiw. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Transactions on Visualization and Computer Graphics*, 2009.

Y. Shinagawa, R. Kawamichi, T. Kunii, and S. Ohwada. Developing surfaces. In *Proceedings SMI. Shape Modeling International 2002*, pages 253–260. IEEE Comput. Soc, 2002.

J. C. Simo and T. A. Laursen. An augmented lagrangian treatment of contact problems involving friction. *Computers and Structures*, 42(1):97–116, 1992.

J. C. Simo, P. Wriggers, and R. L. Taylor. A perturbed lagrangian formulation for the finite element solution of contact problems. *Computer Methods in Applied Mechanics and Engineering*, 50:163–180, 1985.

J. Solomon, E. Vouga, M. Wardetzky, and E. Grinspun. Flexible developable surfaces. *Computer Graphics Forum*, 31, 2012.

J. Stam. Nucleus: Towards a unified dynamics solver for computer graphics. In *IEEE International Conference on Computer-Aided Design and Computer Graphics*, pages 1–11, 2009.

L. Sullivan. The tall office building artistically considered. *Lippincott's Magazine*, 57, 1896.

E. Sultan and A. Boudaoud. Statistics of crumpled paper. *Physical Review Letters*, 96, 2006.

M. Sun and E. Fiume. A Technique for Constructing Developable Surfaces. *Work*, pages 176–185, 1996.

Y. Suris. Hamiltonian methods of runge-kutta type and their variational interpretation. *Matematicheskoe Modelirovanie*, 2(4):78–87, 1990.

T. Tachi. Generalization of rigid foldable quadrilateral mesh origami. *Journal Of The International Association For Shell And Spatial Structures*, 50(October):2287–2294, 2009.

T. Tachi. Simulation of rigid origami. In A. K. Peters, editor, *Origami 4*, page 175. Editorial de la Universitat Politécnica de Valencia., 2009.

T. Tachi. Geometric Considerations for the Design of Rigid Origami Structures. In *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium 2010*, volume 12, pages 458–460, Shanghai, 2010. Elsevier Ltd.

T. Tachi. Origamizing polyhedral surfaces. *IEEE transactions on visualization and computer graphics*, 16(2):298–311, 2010.

T. Tachi. Rigid-Foldable Thick Origami. In P. Wang-Iverson, R. J. Lang, and M. Yim, editors, *Origami 5 Fifth International Meeting of Origami Science Mathematics and Education*, pages 253–264. Taylor & Francis, 2011.

K. T. K. Tang and M. C. M. Chen. Quasi-developable mesh surface interpolation via mesh deformation. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):518–528, 2009.

M. Tang, D. Manocha, and R. Tong. Mccd: Multi-core collision detection between deformable models using front-based decomposition. *Graphical Models*, 72(2):7–23, 2010.

M. Tang, D. Manocha, J. Lin, and R. Tong. Collision-streams: Fast GPU-based collision detection for deformable models. In *I3D '11: Proceedings of the 2011 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 63–70, 2011.

R. L. Taylor and P. Papadopoulos. On a finite element method for dynamic contact/impact problems. *International Journal for Numerical Methods in Engineering*, 36(12):2123–2140, 1993.

B. Thomaszewski and W. Blochinger. Parallel simulation of cloth on distributed memory architectures. *Proc. Eurographics Symp. Parallel Graphics and Visualization*, 2006.

B. Thomaszewski, S. Pabst, and W. Blochinger. Parallel techniques for physically-based simulation on multi-core processor architectures. *Computers and Graphics*, 31:25–40, 2008.

B. Thomaszewski, S. Pabst, and W. Strasser. Asynchronous cloth simulation. *Computer Graphics International*, 2008.

T. Van Mele and P. Block. A novel form finding method for fabric formwork for concrete shells. *J. Int. Assoc. Shell and Spatial Structures*, 52:217–224, 2011.

S. Venkataramani. Lower bounds for the energy in a crumpled elastic sheet – a minimal ridge. *Nonlinearity*, 17(1):301, 2004.

A. P. Veselov. Integrable discrete-time systems and difference operators. *Functional Analysis and Its Applications*, 22(2):83–93, 1988.

E. Vouga, D. Harmon, R. Tamstorf, and E. Grinspun. Asynchronous Variational Contact Mechanics. *Computer Methods in Applied Mechanics and Engineering*, 200:2181–2194, Apr 2011.

E. Vouga, M. Höbinger, J. Wallner, and H. Pottmann. Design of self-supporting surfaces. *ACM Trans. Graphics*, 2012. Proc. SIGGRAPH.

A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Progr.*, 106:25–57, 2006.

C. C. L. Wang. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer*, 26(8-9):167–539, 2004.

C. Wang. Computing length-preserved free boundary for quasi-developable mesh segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):25–36, 2008.

M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun. Discrete Laplace operators: No free lunch. In A. Belyaev and M. Garland, editors, *Symposium on Geometry Processing*, pages 33–37. Eurographics Assoc., 2007.

G. Weiss and P. Furtner. Computer-aided treatment of developable surfaces. *Computers & Graphics*, 12:39–51, 1988.

R. Weller and G. Zachmann. Kinetic separation lists for continuous collision detection of deformable objects. In *Third Workshop in Virtual Reality Interactions and Physical Simulation (Vriphys)*, 2006.

J. M. Wendlandt and J. E. Marsden. Mechanical integrators derived from a discrete variational principle. *Physica D: Nonlinear Phenomena*, 106(3–4):223–246, 1997.

M. West. *Variational integrators*. PhD thesis, California Institute of Technology, 2004.

E. Whiting, J. Ochsendorf, and F. Durand. Procedural modeling of structurally-sound masonry buildings. *ACM Trans. Graphics*, 28(5):#112,1–9, 2009. Proc. SIGGRAPH Asia.

E. Whiting, H. Shin, R. Wang, J. Ochsendorf, and F. Durand. Structural optimization of 3d masonry buildings. *ACM Transactions on Graphics*, 31(6):159:1–159:11, 2012.

T. A. Witten. Stress focusing in elastic sheets. Master's thesis, James Franck Institute, 2006.

P. Wriggers, J. C. Simo, and R. L. Taylor. Penalty and augmented lagrangian formulations for contact problems. In J. Middleton and G. N. Pande, editors, *Proceedings of NUMETA 85 Conference*. Balkema, 1985.

P. Wriggers. Finite element algorithms for contact problems. *Archives of Computational Methods in Engineering*, 2(4):1–49, 1995.

P. Wriggers. *Computational Contact Mechanics*. John Wiley and Sons Ltd, 2002.

S.-E. Yoon, P. Lindstrom, V. Pascucci, and D. Manocha. Cache-oblivious mesh layouts. *ACM Trans. Graph.*, 24(3):886–893, July 2005.

Y. Yoshimura. On the mechanism of buckling of a circular cylindrical shell under axial compression. Technical report, National Advisory Committee for Aeronautics, 1955.

C. Zheng and D. L. James. Toward high-quality modal contact sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4), August 2011.

G. Zhong and J. Marsden. Lie-Poisson Hamilton-Jacobi theory and Lie-Poisson integrators. *Physics Letters A*, 133(3), 1988.